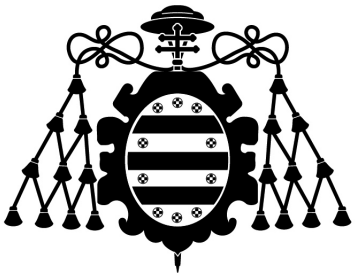


Universidad de Oviedo

Facultad de Ciencias

Grado en Matemáticas

Trabajo de Fin de Grado

La transformada discreta de Fourier. Cálculo óptimo y aplicaciones.

Autor: Miguel de las Heras Rodríguez

Supervisado por:

Santiago Ibáñez Mesa y Pablo Pérez Riera

Curso 2023-2024

Índice general

Resumen

El análisis de Fourier es una potente herramienta matemática cuyo desarrollo ha propiciado grandes avances tecnológicos como el procesamiento de señales de audio, la compresión de imágenes, la espectroscopía o la tomografía computerizada, entre otros.

Desde una perspectiva estrictamente teórica, los dos ingredientes básicos serían los desarrollos en Serie de Fourier y las Transformadas de Fourier, pero estas herramientas están pensadas para funciones definidas sobre la recta real o un intervalo de la recta real. En el análisis de una señal real, uno se enfrenta a un conjunto discreto de datos, y esto motiva la introducción de la Transformada Discreta de Fourier y de la Transformada Rápida de Fourier como método de optimización del cálculo.

El punto de partida de este Trabajo Fin de Grado será el estudio de la Transformada Discreta de Fourier, ocupándonos posteriormente de la rápida. Por supuesto, se incluirán todos los resultados previos que sean precisos, siempre teniendo en cuenta que el estudio general de las Series de Fourier y las Transformadas de Fourier forma parte de los contenidos de algunas de las asignaturas del grado. Entre otros detalles, nos preocuparemos de estudiar el coste computacional de la Transformada Rápida. También incluiremos aplicaciones ilustrativas de la potencia de estas técnicas.

Las principales referencias bibliográficas han sido el libro *Fourier Analysis and its Applications*, de Folland [3]; el libro *Fourier Numerical Analysis*, de Plonka [6]; y el libro *Digital Signal Processing* [7], de Proakis y Manolakis. El primero se ha utilizado para adentrarse en las series y transformadas de Fourier. Para el estudio de la transformada de Fourier Discreta, así como sus propiedades y forma matricial, la principal herramienta ha sido el libro de Plonka. Para abordar la Transformada Rápida nos hemos apoyado de nuevo en el libro de Plonka, y también en el de Proakis y Manolakis. Para resultados clásicos de análisis matemático también tenemos como referencia el libro de Apóstol [1]. En el último capítulo hay información extraída de las páginas web [4] y [5]. Todos los gráficos que aparecen en el trabajo fueron realizados con MATLAB y *Tikz*.

Capítulo 1

Introducción al análisis de Fourier

1.1. Contexto histórico y primeros conceptos

El Análisis de Fourier es una rama de las matemáticas que da respuesta al problema de escribir una función periódica como suma de senos y cosenos; y encuentra su motivación en el afán de entender ciertos problemas físicos, como el de la cuerda vibrante o el de transmisión del calor, abordado por el propio Fourier en 1822 en su obra *Teoría Analítica del Calor*. Posteriormente, todavía en el siglo XIX, esta herramienta fue muy útil para comprender cualquier movimiento ondulatorio, como el de la propagación de la luz o las ondas electromagnéticas. Sin embargo, la física no es el único campo donde el Análisis de Fourier es útil. En un escenario tan alejado de sus orígenes como el de la Teoría de la Probabilidad podemos encontrar una demostración del Teorema Central del Límite en la que se usan técnicas propias del Análisis de Fourier (ver [2]). Actualmente, esta teoría encuentra una de sus principales aplicaciones en el tratamiento de señales, sean de audio, vídeo u otra naturaleza.

Una buena parte de los resultados de este capítulo se estudian en el grado, principalmente en las asignaturas Análisis III y Análisis Funcional. También pueden consultarse en [1], que es una referencia clásica.

Vamos a definir dos espacios vectoriales importantes para el estudio de series y transformadas de Fourier. Antes, recordamos el concepto de función medible:

Definición 1.1. Sean X e Y dos espacios de medida. Una función $f : X \to Y$ se dice medible si para cualquier subconjunto medible $A \subset Y$, el conjunto $f^{-1}(A) \subset X$ también es medible.

En nuestro trabajo $X = \mathbb{R}$ e $Y = \mathbb{C}$, y una función $f : \mathbb{R} \to \mathbb{C}$ se escribe como $f(x) = g(x)+ih(x)$, siendo g y h dos funciones de variable real que toman valores reales. Dotamos a $\mathbb{R}$ de la medida de Lebesgue m, y diremos que f es medible si g y h lo son respecto de m. En las aplicaciones, solo consideraremos funciones $f : \mathbb{R} \to \mathbb{C}$ medibles, y evitaremos entrar en aspectos más propios

de un curso de Teoría de la Medida. Podemos definir el conjunto de las funciones $f : \mathbb{R} \to \mathbb{C}$ medibles y con integral de Lebesgue de $|f|$ finita, que tiene estructura de espacio vectorial y denotaremos por $\mathcal{L}^1(\mathbb{R})$:

$$\mathcal{L}^1(\mathbb{R}) = \left\{ f : \mathbb{R} \to \mathbb{C} \ : \ f \text{ es medible y } \int_{\mathbb{R}} |f| \, dm < \infty \right\}.$$

Lo que se desea es dotar a este espacio de una norma. Dada cualquier función $f \in \mathcal{L}^1(\mathbb{R})$, definimos

$$\|f\|_1 = \int_{\mathbb{R}} |f| \, dm. \tag{1.1}$$

Esta aplicación define una *seminorma* sobre $\mathcal{L}^1(\mathbb{R})$. Añadimos el prefijo "semi" porque falla una condición para poder afirmar que (1.1) es una norma. Toda función f que sea cero salvo en un conjunto de medida nula verificará que $\|f\|_1 = 0$, sin embargo, f no tiene por qué ser la función nula. Este inconveniente se resuelve tomando el espacio cociente que resulta de considerar que dos funciones son equivalentes si coinciden salvo a lo sumo en un conjunto de medida nula. Denotamos este nuevo espacio como $L^1(\mathbb{R})$ y, como es natural, abusando del lenguaje, trataremos a sus elementos como funciones, no como clases de funciones. Ahora, la aplicación definida en (1.1) sí que es una norma en el espacio $L^1(\mathbb{R})$.

Se puede considerar también el siguiente conjunto de funciones medibles:

$$\mathcal{L}^2(\mathbb{R}) = \left\{ f : \mathbb{R} \to \mathbb{C} \ : \ f \text{ es medible y } \int_{\mathbb{R}} |f|^2 dm < \infty \right\}.$$

De nuevo, por los mismos motivos y de forma análoga, trabajamos con un espacio cociente $L^2(\mathbb{R})$, porque así, la aplicación

$$\|f\|_2 = \left(\int_{\mathbb{R}} |f|^2 dm \right)^{1/2} \tag{1.2}$$

define una norma sobre $L^2(\mathbb{R})$. La ventaja que proporciona este nuevo conjunto es su estructura de espacio de Hilbert, ya que la norma definida en (1.2) proviene del producto escalar

$$\langle f, g \rangle = \int_{\mathbb{R}} f \bar{g} \, dm.$$

Introduzcamos ahora el concepto de transformada de Fourier. Dada una función $f \in L^1(\mathbb{R})$, definimos su transformada de Fourier como la función $\hat{f} : \mathbb{R} \to \mathbb{C}$ dada por

$$\hat{f}(\omega) = \int_{\mathbb{R}} f(t) e^{-i\omega t} dt. \tag{1.3}$$

En teoría de la señal, la transformada de Fourier juega un papel fundamental. Permite representar una señal continua definida en el dominio temporal de otra forma equivalente: el dominio frecuencial. Esto se realiza mediante la ponderación de la señal por exponenciales complejas

$e^{-i\omega t}$, siendo ω la frecuencia angular. Esta idea se formaliza mediante la integral (1.3). Además, si la señal cumple algunas propiedades adicionales, podemos recuperarla en el dominio temporal a partir de su expresión en el dominio frecuencial, como muestra el siguiente resultado, cuya prueba se abordó en el contexto de la asignatura *Modelos Matemáticos* de tercer curso y que podemos encontrar en [6].

Teorema 1.1. *Sea $f \in L^1(\mathbb{R}) \cap C(\mathbb{R})$ y supongamos que $\hat{f} \in L^1(\mathbb{R})$. Entonces*

$$f(t) = \frac{1}{2\pi} \int_{\mathbb{R}} \hat{f}(\omega) e^{i\omega t} d\omega.$$

Resulta operativo, entre otras razones por las ventajas que nos da la estructura de espacio de Hilbert, definir transformadas de Fourier en el espacio $L^2(\mathbb{R})$. Esta definición no es directa, pues la integral dada en (1.3) podría no ser finita si $f \in L^2(\mathbb{R})$ pero $f \notin L^1(\mathbb{R})$. Sin embargo, podemos definir la transformada mediante un procedimiento de aproximación basado en la densidad de funciones continuas y con soporte compacto, $C_c^\infty(\mathbb{R})$, en $L^2(\mathbb{R})$. Dada $f \in L^2(\mathbb{R})$ existe una sucesión de funciones $\{f_n\} \subset C_c^\infty$ de manera que

$$\lim_{n\to\infty} \|f_n - f\|_2 = 0.$$

Dado que para cada $n \in \mathbb{N}$, $(f_n)_n$ converge en la norma de $L^2(\mathbb{R})$, entonces es una sucesión de Cauchy en este espacio. Además, al ser continuas y tener soporte compacto, se sigue que $f_n \in L^1(\mathbb{R})$, lo que nos permite definir su transformada de Fourier como en (1.3). Utilizando la identidad de Plancherel [1], tenemos que

$$\|\hat{f}_n - \hat{f}_m\|_2 = \sqrt{2\pi}\|f_n - f_m\|_2,$$

luego $(\hat{f}_n)_n$ también es sucesión de Cauchy en $L^2(\mathbb{R})$. Dado que este espacio es de Hilbert, la sucesión es convergente. Este límite es lo que tomamos como definición de transformada de Fourier de la función inicial $f \in L^2(\mathbb{R})$, denotándola como $\hat{f}$. También existe un resultado análogo al Teorema 1.1 para funciones de este nuevo espacio:

Teorema 1.2. *Sea $f \in L^2(\mathbb{R})$ con $\hat{f} \in L^1(\mathbb{R})$. Entonces, se cumple que*

$$f(t) = \frac{1}{2\pi} \int_{\mathbb{R}} \hat{f}(\omega) e^{i\omega t} d\omega, \tag{1.4}$$

en casi todo punto. Es decir, la expresión anterior es cierta salvo a lo sumo en un conjunto de medida nula. Si además f es continua, entonces la identidad (1.4) es válida en toda la recta real.

[1] Si $f, h \in L^1(\mathbb{R}) \cap L^2(\mathbb{R})$, entonces

$$\int_{\mathbb{R}} f(t)\bar{h}(t)\, dt = \frac{1}{2\pi} \int_{\mathbb{R}} \hat{f}(\omega)\bar{\hat{h}}(\omega)\, d\omega.$$

Ahora, vamos a considerar que la función $f : \mathbb{R} \to \mathbb{C}$ es periódica. Sin pérdida de generalidad, podemos considerar que su periodo es 2π, ya que si tenemos una función T-periódica $h(x)$, la función $g(x) = h(\frac{T}{2\pi}x)$ es 2π-periódica. Además, podemos considerar una identificación con funciones definidas sobre la circunferencia: Para toda función $f : \mathbb{R} \to \mathbb{C}$ 2π-periódica, podemos introducir una función $F : \mathbb{T} \to \mathbb{C}$, donde $\mathbb{T}$ denota el toro unidimensional, es decir, la circunferencia, definida por $F(e^{it}) = f(t)$ para cada $t \in \mathbb{R}$. Recíprocamente, dada una función $F : \mathbb{T} \to \mathbb{C}$ podemos definir una función 2π-periódica $f : \mathbb{R} \to \mathbb{C}$, con $f(t) = F(e^{it})$ para cada $t \in \mathbb{R}$. Consideramos entonces el espacio

$$L^2(\mathbb{T}) = \left\{ f : \mathbb{R} \to \mathbb{C} : \ f \text{ es medible, } 2\pi\text{-periódica y } \int_{-\pi}^{\pi} |f(t)|^2 \, dt < \infty \right\}.$$

Este espacio es de Hilbert con el producto escalar

$$\langle f, g \rangle = \int_{-\pi}^{\pi} f(t)\bar{g}(t) \, dt.$$

Las razones para que los límites de integración sean $-\pi$ y π son puramente operativas. Podríamos considerar la integral en cualquier otro intervalo de longitud 2π, en virtud del siguiente resultado:

Proposición 1.3. *Si f es periódica de periodo T, entonces la función $g : \mathbb{R} \to \mathbb{C}$, definida por $g(a) = \int_a^{a+T} f(t) \, dt$, es constante.*

Demostración. Podemos escribir g como

$$g(a) = \int_0^{a+T} f(x)dx - \int_0^a f(x)dx.$$

Por el Teorema Fundamental del Cálculo, $g'(a) = f(a + T) - f(a)$ y, teniendo en cuenta la periodicidad de f, tenemos que $g'(a) = 0$ para todo a y entonces g es constante. □

Dada $f \in L^2(\mathbb{T})$, para cada $k \in \mathbb{Z}$ definimos el coeficiente de Fourier k-ésimo de f como

$$c_k = \frac{1}{2\pi} \int_{-\pi}^{\pi} f(t)e^{-ikt}dt. \tag{1.5}$$

Para cada k, el coeficiente de fourier c_k está bien definido. En efecto, sea $f \in L^2(\mathbb{T})$. Se puede probar, apoyándonos en la desigualdad de Cauchy-Schwarz [2], que $f \in L^1(\mathbb{T})$, donde

$$L^1(\mathbb{T}) = \left\{ f : \mathbb{R} \to \mathbb{C} : \ f \text{ es medible}, 2\pi\text{-periódica y } \int_{-\pi}^{\pi} |f(t)| \, dt < \infty \right\}.$$

[2]Dadas dos funciones f y g de $L^2(\mathbb{T})$, se cumple que $|\langle f, g\rangle| = |\int_{-\pi}^{\pi} f(t)\overline{g(t)}dt| \leq \sqrt{\|f\|_2}\sqrt{\|g\|}_2$

Así,

$$|c_k| \leq \frac{1}{2\pi}\int_{-\pi}^{\pi} |f(t)e^{-ikt}|dt \leq \frac{1}{2\pi}\int_{-\pi}^{\pi} |f(t)|dt < \infty.$$

Para cada $n \in \mathbb{N}$, consideremos el operador $S_n : L^2(\mathbb{T}) \to L^2(\mathbb{T})$ con

$$S_n f(e^{ikt}) := \sum_{k=-n}^{n} c_k e^{ikt}, \tag{1.6}$$

para cada $t \in \mathbb{R}$.

Recordamos el siguiente resultado, cuya demostración forma parte de los contenidos de la asignatura *Análisis III* (ver también [6]).

Teorema 1.4. *Toda función $f \in L^2(\mathbb{T})$ tiene una única representación de la forma*

$$f(e^{ikt}) = \sum_{k\in\mathbb{Z}} c_k e^{ikt}, \tag{1.7}$$

con coeficientes c_k dados por (1.5) y donde la sucesión $\{S_n f\}_{n\in\mathbb{N}}$ converge en la norma de $L^2(\mathbb{T})$ a f, es decir,

$$\lim_{n\to\infty} \|S_n f - f\|_2 = 0.$$

Se sigue a partir de este teorema que cualquier $f \in L^2(\mathbb{T})$ puede identificarse con su desarrollo de Fourier (serie de Fourier) (1.7) en la norma de L^2.

En la asignatura *Análisis III*, también se estudiaron otros resultados que nos aportan información acerca de la convergencia puntual y uniforme. Por ejemplo, si tenemos una función periódica regular a trozos, hay convergencia puntual de su serie de Fourier en aquellos puntos en los que la función es continua. También se cumple que si la función es continua con derivada continua a trozos, entonces su serie de Fourier converge uniformemente a la función. El siguiente ejemplo (ver [6]) ilustra un caso en el que podemos calcular analíticamente los coeficientes de Fourier y observar cómo la función se corresponde con su desarrollo en serie de Fourier.

Ejemplo 1.1 Sea f la extensión 2π-periódica de la función de variable real $f(x) = e^{-x}$ si $x \in (-\pi, \pi)$ y $f(\pm\pi) = \cosh\pi$. Entonces, escribimos los coeficientes de Fourier como sigue:

$$c_k = \frac{1}{2\pi}\int_{-\pi}^{\pi} e^{-(1+ik)x}dx = -\frac{1}{2\pi(1+ik)}(e^{-(1+ik)\pi} - e^{(1+ik)\pi}) = \frac{(-1)^k \sinh\pi}{(1+ik)\pi}$$

En la Figura 2.2 representamos las sumas parciales $S_n f$ para $n = 8$ y $n = 16$, respectivamente, y mostramos cómo éstas se aproximan a la función f.

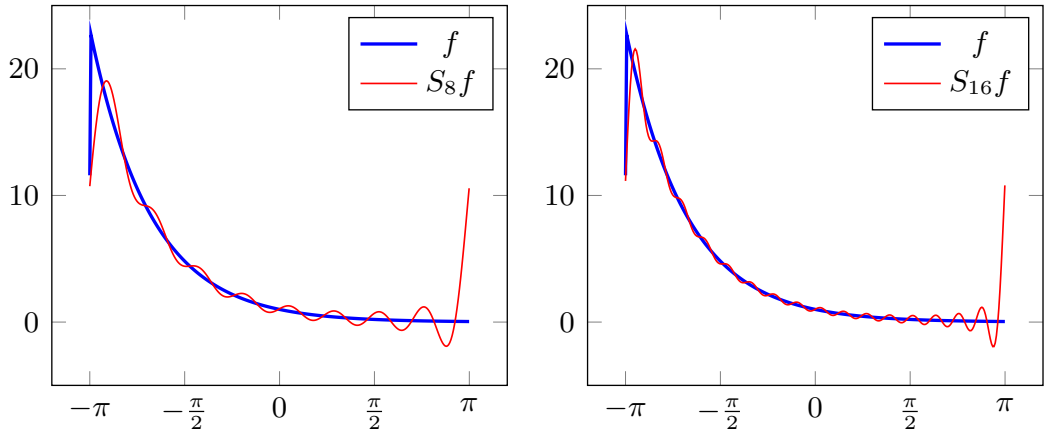


Figura 1.1: La suma (1.6) para $n = 8$ (izquierda) y $n = 16$ (derecha)

A continuación, vamos a ver cómo podríamos escribir la serie de Fourier como una combinación lineal de senos y cosenos. Como el coseno es par y el seno es impar, dada cualquier $f \in L^1(\mathbb{T})$, resulta:

$$S(t) = \sum_{k\in\mathbb{Z}} c_k(\cos(kt) + i\,\operatorname{sen}(kt)) = c_0 + \sum_{n=1}^{\infty}\big[(c_n + c_{-n})\cos(nt) + i(c_n - c_{-n})\operatorname{sen}(nt)\big].$$

Teniendo en cuenta que

$$c_n + c_{-n} = \frac{1}{2\pi}\int_{-\pi}^{\pi} f(t)(e^{-int} + e^{int})dt = \frac{1}{\pi}\int_{-\pi}^{\pi} f(t)\cos(nt)\,dt,$$

$$i(c_n - c_{-n}) = \frac{i}{2\pi}\int_{-\pi}^{\pi} f(t)(e^{-int} - e^{int})dt = \frac{1}{\pi}\int_{-\pi}^{\pi} f(t)\operatorname{sen}(nt)\,dt,$$

obtenemos

$$S(t) = \frac{a_0}{2} + \sum_{n=1}^{\infty}\big(a_n\cos(nt) + b_n\operatorname{sen}(nt)\big), \tag{1.8}$$

donde

$$a_n = \frac{1}{\pi}\int_{-\pi}^{\pi} f(t)\cos(nt)\,dt,$$
$$b_n = \frac{1}{\pi}\int_{-\pi}^{\pi} f(t)\sin(nt)\,dt,$$
$$a_0 = 2c_0 = \frac{1}{\pi}\int_{-\pi}^{\pi} f(t)\,dt.$$

La forma (1.8) recibe el nombre de forma real de la serie de Fourier. También se puede definir de partida la serie de Fourier de esta manera y obtener a partir de ella un desarrollo de Fourier con exponenciales.

Observación 1.1. Como se vio en *Análisis III* (consultar también [1]), también se pueden definir las series de Fourier en intervalos acotados y posteriormente realizar la generalización para funciones periódicas.

La teoría de series y transformadas de Fourier es un tema muy amplio por sí mismo y sobre ella podrían elaborarse varios trabajos Fin de Grado. Nosotros partiremos de un número finito de puntos en los que conocemos el valor de la señal f que queramos estudiar. Para ello, nuestra herramienta será la transformada de Fourier discreta, que para cada punto de la señal nos aporta el contenido frecuencial de la misma en ese punto. También estudiaremos la relación entre la transformada de Fourier discreta y las series y transformadas de Fourier en un caso concreto sencillo. A su vez, explicaremos e implementaremos algoritmos rápidos para el cálculo de la transformada discreta y analizaremos su coste computacional. El lector podría plantearse la cuestión de si es suficiente un número finito de puntos para obtener información detallada de una señal continua y la respuesta es que sí; lo veremos en el último capítulo.

Capítulo 2

La transformada discreta de Fourier

En el capítulo anterior hemos presentado un marco general para el análisis de Fourier. De manera concisa, el análisis de Fourier aporta una nueva perspectiva para entender las funciones, al representarlas en términos de frecuencia, en lugar de en términos de tiempo o espacio como es más habitual. En la práctica, rara vez vamos a disponer de una expresión explícita de una función y, aún en ese caso, el cálculo de los coeficientes y las transformadas de Fourier involucran integrales que serán muy difíciles de resolver, o incluso imposibles. Ante este problema, la transformada de Fourier discreta es una herramienta muy poderosa. Este capítulo va a centrarse en definir el concepto de transformada discreta de Fourier (DFT), así como su forma matricial, recogiendo propiedades sobre la matriz de Fourier, que seguidamente usaremos para probar resultados acerca de la transformada de Fourier discreta. Como ya comentamos al principio del trabajo, la principal referencia para este capítulo es el libro de Plonka *Fourier Numerical Analysis* [6].

2.1. Introducción a la transformada discreta de Fourier

La transformada discreta de Fourier es una herramienta para extraer información frecuencial de una señal finita. Una señal finita de tamaño N es un vector $\mathbf{f} \in \mathbb{C}^N$, cuyas componentes denotaremos indistintamente por $\mathbf{f}[n]$ o por $\mathbf{f}_n$, para $n = 0, 1, \ldots, N-1$. En nuestras aplicaciones, este vector vendrá dado por la evaluación de una señal continua en una colección finita de puntos. El número de puntos que tomaremos vendrá dado por la frecuencia de muestreo, f_s, es decir, el número de muestras recogidas por unidad de tiempo. Habitualmente, la variable independiente es el tiempo y se mide en segundos.

Sabemos que $\mathbb{C}^N$ es un espacio vectorial euclídeo de dimensión N con el producto escalar

$$\langle \mathbf{f}, \mathbf{h} \rangle = \sum_{n=0}^{N-1} \mathbf{f}[n]\bar{\mathbf{h}}[n],$$

donde $\overline{\mathbf{z}}$ denota el complejo conjugado de $\mathbf{z} \in \mathbb{C}^N$. En la siguiente definición se introduce una notación muy recurrente a lo largo del texto:

Definición 2.1. Dado $N \in \mathbb{N}$, se define la raíz N-ésima de la unidad como cualquier número complejo z que verifique la identidad

$$z^N = 1.$$

Utilizando la notación

$$\omega_N = e^{-i2\pi/N} = \cos\frac{2\pi}{N} - i\,\mathrm{sen}\,\frac{2\pi}{N}, \tag{2.1}$$

Notamos que

$$(\omega_N^k)^N = e^{-2\pi ikN/N} = e^{-2\pi ik} = 1, \tag{2.2}$$

para $k = 0, 1, \ldots, N-1$. Así, $(\omega_N^k)_{k=0}^{N-1}$ son las N raíces de la unidad.

El siguiente teorema proporciona una base ortogonal para el espacio de señales finitas de dimensión N y motiva la definición de la transformada discreta de Fourier que le sigue.

Teorema 2.1. *Para cada $k = 0, 1, \ldots, N-1$, sea $\mathbf{e}_k = (\mathbf{e}_k[0], \mathbf{e}_k[1], \ldots, \mathbf{e}_k[N-1]) \in \mathbb{C}^N$, con $\mathbf{e}_k[n] = \omega_N^{nk}$. Entonces, la familia $\{\mathbf{e}_k\}_{k=0}^{N-1}$ forma una base ortogonal del espacio de señales finitas $\mathbb{C}^N$.*

Demostración. Para $j, k = 0, 1, \ldots, N-1$, tenemos

$$\langle \mathbf{e}_j, \mathbf{e}_k \rangle = \sum_{n=0}^{N-1} e^{-2\pi ijn/N} e^{2\pi ikn/N} = \sum_{n=0}^{N-1} e^{2\pi in(k-j)/N}.$$

Si $j = k$, entonces $\|\mathbf{e}_j\|^2 = \langle \mathbf{e}_j, \mathbf{e}_j \rangle = N$.

Supongamos ahora que $j \neq k$ y denotemos $z = e^{2\pi i(k-j)/N}$, así $z \neq 1$ y además

$$z^N = e^{2\pi i(k-j)} = \cos\big(2\pi(k-j)\big) + i\,\mathrm{sen}\big(2\pi(k-j)\big) = 1,$$

debido a que $(k-j)$ es un número entero. Entonces

$$\langle \mathbf{e}_j, \mathbf{e}_k \rangle = 1 + z + \ldots + z^{N-1} = \frac{z^N - 1}{z - 1} = 0,$$

Hemos probado que $\{\mathbf{e}_k\}_{k=0}^{N-1}$ es una familia ortogonal de N vectores contenidos en $\mathbb{C}^N$, luego forman una base. □

Definición 2.2. Sea $\mathbf{f}$ una señal finita de dimensión N. La transformada de Fourier discreta (DFT, por sus siglas en inglés) de $\mathbf{f}$ es un vector $\hat{\mathbf{f}} \in \mathbb{C}^N$ cuyas componentes son

$$\hat{\mathbf{f}}[k] = \langle \mathbf{f}, \mathbf{e}_k \rangle = \sum_{n=0}^{N-1} \mathbf{f}[n]\, e^{-2\pi ikn/N} = \sum_{n=0}^{N-1} \mathbf{f}[n]\, \omega_N^{nk}, \tag{2.3}$$

para $k = 0, 1, \ldots, N-1$.

La descomposición de $\mathbf{f}$ en la base $\{\mathbf{e}_k\}_{k=0}^{N-1}$ nos proporciona una fórmula para la *transformada inversa de Fourier discreta*:

$$\mathbf{f}[n] = \sum_{k=0}^{N-1} \frac{\langle \mathbf{f}, \mathbf{e}_k \rangle}{\|\mathbf{e}_k\|^2} \mathbf{e}_k[n] = \frac{1}{N} \sum_{k=0}^{N-1} \hat{\mathbf{f}}[k] e^{2\pi i k n/N} = \frac{1}{N} \sum_{k=0}^{N-1} \hat{\mathbf{f}}[k] \omega_N^{-nk}, \tag{2.4}$$

para $n = 0, 1, \ldots, N-1$.

2.2. Comprendiendo la ecuación de la DFT

En esta sección, entenderemos el significado de la expresión (2.3). Utilizando la identidad de Euler, la expresión toma la forma:

$$\hat{\mathbf{f}}[k] = \sum_{n=0}^{N-1} \mathbf{f}[n] \Big(\cos \frac{2\pi k n}{N} - i \operatorname{sen} \frac{2\pi k n}{N} \Big).$$

Esta representación facilita la comprensión de la Transformada Discreta de Fourier (DFT). Para cada $k = 0, 1, \ldots, N-1$, la DFT caracteriza el contenido frecuencial en dicho punto mediante la suma, punto por punto, del producto del valor de la señal y una onda sinusoidal de la forma $\cos\theta - i \operatorname{sen}\theta$, donde θ representa la fase angular de la onda. Esto permite conocer la contribución de cada frecuencia al espectro de la señal, permitiendo el análisis de la misma en el dominio de frecuencia.

Por otro lado, la fórmula (2.4) nos permite reconstruir la señal original a partir de sus componentes frecuenciales. Esta propiedad es fundamental en la recuperación de datos, ya que nos permite trabajar de manera eficiente en el dominio de frecuencias y luego reconstruir la señal en el dominio de tiempos.

La Transformada de Fourier Discreta resulta ser una herramienta muy poderosa en numerosos campos como la ingeniería, las ciencias de computación, la medicina o el procesamiento de señales.

2.3. Forma matricial de la DFT

Vamos a considerar $\mathbf{f}$ y $\hat{\mathbf{f}}$ como vectores columna. Podemos pensar en una expresión matricial de la DFT:

$$\hat{\mathbf{f}} = \mathbf{F}_N \mathbf{f},$$

siendo $\mathbf{F}_N$ una matriz de tamaño $N \times N$. En realidad, estamos viendo la DFT como un endomorfismo de $\mathbb{C}^N$, cuya matriz asociada es $\mathbf{F}_N$. Utilizando la misma notación introducida en la

Definición 2.1, podemos escribir la matriz de Fourier como sigue:

$$\mathbf{F}_N = \begin{bmatrix} 1 & 1 & 1 & \dots & 1 \\ 1 & \omega_N & \omega_N^2 & \dots & \omega_N^{N-1} \\ 1 & \omega_N^2 & \omega_N^{2\cdot 2} & \dots & \omega_N^{2(N-1)} \\ \vdots & \vdots & \vdots & & \vdots \\ 1 & \omega_N^{N-1} & \omega_N^{2(N-1)} & \dots & \omega_N^{(N-1)^2} \end{bmatrix}.$$

Supongamos que $kn \equiv m_2$ mód N, es decir, que $kn = m_2 + m_1 N$, con $m_1 \in \mathbb{Z}$. Así,

$$\omega_N^{kn} = (\omega_N^N)^{m_1}\omega_N^{m_2} = 1^{m_1}\omega_N^{m_2}. \tag{2.5}$$

Usando esta relación, podemos reescribir la matriz de Fourier como sigue:

$$\mathbf{F}_N = \begin{bmatrix} 1 & 1 & 1 & \dots & 1 \\ 1 & \omega_N & \omega_N^2 & \dots & \omega_N^{N-1} \\ 1 & \omega_N^2 & \omega_N^{2\cdot 2} & \dots & \omega_N^{N-2} \\ \vdots & \vdots & \vdots & & \vdots \\ 1 & \omega_N^{N-1} & \omega_N^{N-2} & \dots & \omega_N \end{bmatrix}.$$

Notamos que se trata de una matriz simétrica. Sus columnas son los vectores $\mathbf{e}_k$ del Teorema 2.1 y por la demostración del teorema en cuestión, se tiene que

$$\mathbf{F}_N^\top \overline{\mathbf{F}}_N = N\mathbf{I}_N,$$

donde $\mathbf{I}_N$ denota la matriz identidad $N \times N$. La matriz de Fourier reescalada, $\frac{1}{\sqrt{N}}\mathbf{F}_N$, es unitaria.

Observación 2.1. El coste computacional del cálculo de la DFT tiene un orden de $O(N^2)$. Uno puede imaginarse lo que esto supondría cuando trabajamos con muestras grandes. Sin embargo, este problema tiene solución. Usando las simetrías de la DFT, este coste puede reducirse al orden $O(N\log(N))$. Para comprender lo que significa este ahorro, suponemos que tenemos una muestra de 64 puntos, que es una cantidad muy pequeña. Para $N = 64$, se tiene $N^2 = 4096$, pero $N\log(N) \approx 115$. Abordaremos la cuestión de la eficiencia computacional en el siguiente capítulo sobre transformada rápida de Fourier (FFT).

Observación 2.2. Sea $N \in \mathbb{N}$, con $N > 1$. Para cada $k \in \mathbb{Z}$, se puede definir el valor

$$\hat{\mathbf{f}}[k+N] = \sum_{n=0}^{N-1} \mathbf{f}[n]\omega_N^{nk},$$

ya que

$$\omega_N^{n(k+N)} = \omega_N^{nk} \cdot 1 = \omega_N^{nk}.$$

Claramente la sucesión $\hat{\mathbf{f}} = \big(\hat{\mathbf{f}}[k]\big)_{k\in\mathbb{Z}}$, es N- periódica. De forma análoga, usando la expresión (2.4), se obtiene que la sucesión $f = (f[n])_{n\in\mathbb{Z}}$ definida por

$$\mathbf{f}[n] = \frac{1}{N}\sum_{k=0}^{N-1} \hat{\mathbf{f}}[k]\omega_N^{-nk},$$

es también N- periódica. Acabamos de ver que la Definición 2.2 puede extenderse para sucesiones N- periódicas $\mathbf{f} = \big(\mathbf{f}[n]\big)_{n\in\mathbb{Z}}$. Esta extensión también se conoce como serie de Fourier de la sucesión N- periódica $\mathbf{f}$.

Como consecuencia de la observación anterior, para cualquier $N \in \mathbb{N}$ se podría definir definir la DFT N- dimensional a partir de cualquier subvector N- dimensional de la sucesión N-periódica $\big(\mathbf{f}[n]\big)_{n\in\mathbb{N}}$. Por ejemplo, si escogemos $\big(\mathbf{f}[n]\big)_{n=-N/2}^{N/2-1}$,

$$\sum_{n=-N/2}^{N/2-1} \mathbf{f}[n]\omega_N^{nk} = \sum_{n=1}^{N/2} \mathbf{f}[N-n]\omega_N^{(N-n)k} + \sum_{n=0}^{N/2-1} \mathbf{f}[n]\omega_N^{nk} = \sum_{n=0}^{N-1} \mathbf{f}[n]\omega_N^{nk},$$

para todo $k \in \mathbb{Z}$, obteniendo así la expresión dada en 2.2.

2.3.1. Propiedades de la matriz de Fourier

Teorema 2.2. *Para cualquier $N \in \mathbb{N}$, la matriz de Fourier $\mathbf{F}_N$ es invertible y además su inversa es*

$$\mathbf{F}_N^{-1} = \frac{1}{N}\overline{\mathbf{F}}_N. \tag{2.6}$$

Demostración. Sea $\mathbf{f} \in \mathbb{C}^N$, de manera que $\hat{\mathbf{f}} = \mathbf{F}_N\mathbf{f}$. Esto es equivalente a que para cada $k = 0, 1, \ldots, N-1$,

$$\hat{\mathbf{f}}_k = \sum_{n=0}^{N-1} \mathbf{f}_n\omega_N^{nk}.$$

Como ya se hizo en (2.4), podemos recuperar la expresión de $\mathbf{f}_n$ para $n = 0, 1, \ldots, N-1$. En particular,

$$\mathbf{f}_n = \frac{1}{N}\sum_{k=0}^{N-1} \hat{\mathbf{f}}_k\omega_N^{-nk} = \frac{1}{N}\sum_{k=0}^{N-1} \hat{\mathbf{f}}_k\overline{\omega_N^{nk}}.$$

Escribiendo esta última expresión en forma matricial, lo que se tiene es que

$$\mathbf{f} = \frac{1}{N}\mathbf{G}_N\hat{\mathbf{f}},$$

siendo

$$\mathbf{G}_N = \begin{bmatrix} 1 & 1 & 1 & \dots & 1 \\ 1 & \omega_N^{-1} & \omega_N^{-2} & \dots & \omega_N^{-(N-1)} \\ 1 & \omega_N^{-2} & \omega_N^{-2\cdot 2} & \dots & \omega_N^{-(N-2)} \\ \vdots & \vdots & \vdots & & \vdots \\ 1 & \omega_N^{-(N-1)} & \omega_N^{-(N-2)} & \dots & \omega_N^{-1} \end{bmatrix} = \overline{\mathbf{F}_N}.$$

En definitiva, hemos probado que

$$\mathbf{F}_N^{-1} = \frac{1}{N}\mathbf{G}_N = \frac{1}{N}\overline{\mathbf{F}_N}. \qquad \square$$

Definición 2.3. Para cada $j \in \mathbb{Z}$, definimos la delta de Kronecker como

$$\delta_j = \begin{cases} 1 \text{ si } j = 0, \\ 0 \text{ si } j \neq 0. \end{cases}$$

Para $j \in \mathbb{Z}$ y $N \in \mathbb{N}$ se denota por

$$j \text{ mód } N \in \{0, 1, \dots, N-1\},$$

al resto no negativo de dividir j por N.

El siguiente lema resulta de especial utilidad para probar resultados posteriores:

Lema 2.3. *Sea $N \in \mathbb{N}$ arbitrario. Para cada $j \in \mathbb{Z}$, se cumple la siguiente propiedad:*

$$\sum_{k=0}^{N-1} \omega_N^{jk} = N\delta_{j \text{ mód } N}. \tag{2.7}$$

Demostración. Sean $j \in \mathbb{Z}$ y $N \in \mathbb{N}$ arbitrarios. Si j mód $N = 0$, entonces $\delta_{j \text{ mód } N} = 1$ y $j = \alpha N$ para algún $\alpha \in \mathbb{Z}$. Así,

$$\omega_N^j = \left(\omega_N^N\right)^\alpha = 1.$$

Por tanto,

$$N\delta_{j \text{ mód } N} = N = \sum_{k=0}^{N-1} \omega_N^{jk}.$$

Si j mód $N \neq 0$, $j = \alpha N + m$ para ciertos $\alpha \in \mathbb{Z}$ y $m \in \{1, \dots, N-1\}$ y por lo tanto

$$\omega_N^j = \omega_N^{\alpha N + m} = \left(\omega_N^N\right)^\alpha \omega_N^m = \omega_N^m \neq 1.$$

Así, teniendo en cuenta esto, si $x \neq 1$, se cumple

$$\sum_{k=0}^{N-1} x^k = \frac{x^N - 1}{x - 1}.$$

Se sigue entonces que

$$\sum_{k=0}^{N-1} \omega_N^{jk} = \sum_{k=0}^{N-1} \left(\omega_N^m\right)^k = 0.$$

Así, como $\delta_{j \bmod N} = 0$ cuando $j \bmod N \neq 0$, obtenemos (2.7) para este caso y la prueba concluye. □

Lema 2.4. *La matriz de Fourier* $\mathbf{F}_N$ *cumple*

$$\mathbf{F}_N^2 = N\mathbf{J}'_N, \tag{2.8}$$

y también

$$\mathbf{F}_N^4 = N^2\mathbf{I}_N, \tag{2.9}$$

donde $\mathbf{J}'_N$ *es la matriz de volteo*

$$\mathbf{J}'_N = \left(\delta_{(j+k) \bmod N}\right)_{j,k=0}^{N-1} = \begin{bmatrix} 1 & 0 & 0 & \dots & 0 & 0 \\ 0 & 0 & 0 & \dots & 0 & 1 \\ \vdots & \vdots & \vdots & & \iddots & \vdots \\ \vdots & \vdots & \vdots & \iddots & & \vdots \\ 0 & 0 & 1 & \dots & 0 & 0 \\ 0 & 1 & \dots & \dots & 0 & 0 \end{bmatrix}.$$

Además,

$$\mathbf{F}_N^{-1} = \frac{1}{N}\mathbf{J}'_N\mathbf{F}_N = \frac{1}{N}\mathbf{F}_N\mathbf{J}'_N. \tag{2.10}$$

Demostración. La expresión (2.8) se sigue fácilmente del Lema 2.3, pues $\mathbf{F}_N^2 = \left(\rho_{j,l}\right)_{j,l=0}^{N-1}$, donde

$$\rho_{j,l} = \sum_{k=0}^{N-1} \omega_N^{jk}\omega_N^{kl} = \sum_{k=0}^{N-1} \omega_N^{(j+l)k} = N\delta_{(j+l) \bmod N}.$$

Como $(\mathbf{J}'_N)^2 = \mathbf{I}_N$, tenemos que

$$\mathbf{F}_N^4 = \mathbf{F}_N^2\mathbf{F}_N^2 = (N\mathbf{J}'_N)(N\mathbf{J}'_N) = N^2(\mathbf{J}'_N)^2 = N^2\mathbf{I}_N,$$

por lo que también

$$\mathbf{F}_N^{-1} = \frac{1}{N^2}\mathbf{F}_N^3 = \frac{1}{N^2}(\mathbf{F}_N)^2\mathbf{F}_N = \frac{1}{N^2}\mathbf{F}_N(\mathbf{F}_N)^2 = \frac{1}{N}\mathbf{F}_N\mathbf{J}'_N = \frac{1}{N}\mathbf{J}'_N\mathbf{F}_N. \qquad \square$$

Observación 2.3. El nombre de matriz de *volteo* se debe a que al actuar esta matriz sobre un vector $\mathbf{a} = (a_0, a_1, \ldots, a_{N-1})^\intercal$, voltea todos sus elementos dejando el primero fijo, es decir,

$$\mathbf{J}'_N\mathbf{a} = (a_0, a_{N-1}, \ldots, a_2, a_1)^\intercal.$$

Observación 2.4. La fórmula (2.10) muestra que la matriz de Fourier inversa se puede expresar como el producto de la matriz de volteo y la propia matriz de Fourier, dividido por N. Por tanto, un algoritmo que calcule la DFT de un vector nos servirá para calcular la DFT inversa con cambios mínimos. En particular, cada vez que aparezca un factor de rotación ω_N, habría que sustituirlo por su conjugado, es decir, ω_N^{-1}, y dividir el resultado final por N, según (2.6).

2.4. La DFT y convoluciones cíclicas. Propiedades básicas de la DFT

Definición 2.4. La convolución cíclica de dos vectores $\mathbf{a} = (a_k)_{k=0}^{N-1}$, $\mathbf{b} = (b_k)_{k=0}^{N-1} \in \mathbb{C}^N$, que se denota $\mathbf{a} * \mathbf{b}$, se define como el vector $\mathbf{c} = (c_n)_{n=0}^{N-1}$ con componentes

$$c_n = \sum_{k=0}^{N-1} a_k b_{(n-k) \bmod N} = \sum_{k=0}^{n} a_k b_{n-k} + \sum_{k=n+1}^{N-1} a_k b_{N+n-k},$$

para $n = 0, 1, \ldots, N-1$.

La convolución cíclica en $\mathbb{C}^N$ es una operación conmutativa, asociativa y distributiva. Además, existe un elemento unidad $\mathbf{b}_0 = (\delta_{j \bmod N})_{j=0}^{N-1} = (1, 0, \ldots, 0)^\intercal$, al que nos referiremos como *vector de impulso*.

Definimos a continuación la *matriz de desplazamiento positivo*, $\mathbf{V}_N$ como

$$\mathbf{V}_N = (\delta_{(j-k-1) \bmod N})_{j,k=0}^{N-1} = \begin{bmatrix} 0 & 0 & \ldots & 0 & 1 \\ 1 & 0 & \ldots & 0 & 0 \\ 0 & 1 & \ldots & 0 & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & \ldots & 1 & 0 \end{bmatrix}.$$

La multiplicación de esta matriz por un vector $\mathbf{a} = (a_k)_{k=0}^{N-1}$ resulta ser el vector $\mathbf{a}$ con todos sus

elementos desplazados hacia la derecha una posición, de ahí el nombre de matriz de desplazamiento positivo:

$$\mathbf{V}_N\mathbf{a} = (a_{(j-1) \bmod N})_{j=0}^{N-1} = (a_{N-1}, a_0, a_1, \ldots, a_{N-2})^\intercal.$$

Si multiplicamos la matriz $\mathbf{V}_N$ por sí misma, obtenemos

$$\mathbf{V}_N^2 = (\delta_{(j-k-2) \bmod N})_{j,k=0}^{N-1} = \begin{bmatrix} 0 & 0 & \ldots & 1 & 0 \\ 0 & 0 & \ldots & 0 & 1 \\ 1 & 0 & \ldots & 0 & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 1 & \ldots & 0 & 0 \end{bmatrix}$$

y

$$\mathbf{V}_N^2\mathbf{a} = (a_{(j-2) \bmod N})_{j=0}^{N-1} = (a_{N-2}, a_{N-1}, a_0, \ldots, a_{N-3}).$$

Además, se puede comprobar que $\mathbf{V}_N^N = \mathbf{I}_N$, de donde se sigue que

$$\mathbf{V}_N^{-1} = \mathbf{V}_N^{N-1} = \begin{bmatrix} 0 & 1 & 0 & \ldots & 0 \\ 0 & 0 & 1 & \ldots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \ldots & 1 \\ 1 & 0 & 0 & \ldots & 0 \end{bmatrix}.$$

Notamos entonces que $\mathbf{V}_N^{-1} = \mathbf{V}_N^\intercal$. Nos referiremos a esta matriz con el nombre de *matriz de desplazamiento negativo*, ya que el vector

$$\mathbf{V}_N^{-1}\mathbf{a} = (a_{(j+1) \bmod N})_{j=0}^{N-1} = (a_1, a_2, \ldots, a_{N-1}, a_0)^\intercal,$$

es el vector $\mathbf{a}$ con todos sus elementos desplazados una posición hacia la izquierda, excepto el primero, que ocupa la última posición. Del mismo modo, podemos considerar la *matriz de diferencia cíclica* $\mathbf{I}_N - \mathbf{V}_N$:

$$(\mathbf{I}_N - \mathbf{V}_N)\mathbf{a} = (a_j - a_{(j-1) \bmod N})_{j=0}^{N-1} = (a_0 - a_{N-1}, a_1 - a_0, \ldots, a_{N-1} - a_{N-2})^\intercal.$$

Observamos también que

$$\mathbf{I}_N + \mathbf{V}_N + \mathbf{V}_N^2 + \cdots + \mathbf{V}_N^{N-1} = (1)_{j,k=0}^{N-1}.$$

Nuestro objetivo ahora es caracterizar todos los endomorfismos $\mathbf{H}_N$ de $\mathbf{C}^N$ que son invariantes

por desplazamiento, es decir,

$$\mathbf{H}_N(\mathbf{V}_N\mathbf{a}) = \mathbf{V}_N(\mathbf{H}_N\mathbf{a}), \tag{2.11}$$

para todo $\mathbf{a} \in \mathbb{C}^N$, ya que éstos juegan un papel muy importante en el filtrado de señales. Notamos que de (2.11) se sigue fácilmente mediante un razonamiento inductivo que, para todo $k = 0, 1, \ldots, N-1$,

$$\mathbf{H}_N\mathbf{V}_N^k = \mathbf{V}_N^k\mathbf{H}_N.$$

Lema 2.5. *Cualquier endomorfismo invariante por desplazamiento $\mathbf{H}_N$ de $\mathbb{C}^N$ cumple que, para todo $\mathbf{a} \in \mathbb{C}^N$,*

$$\mathbf{H}_N\mathbf{a} = \mathbf{a} * \mathbf{h},$$

donde $\mathbf{h} = \mathbf{H}_N\mathbf{b}_0$ es el vector de respuesta al vector de impulso $\mathbf{b}_0$ definido al comienzo de la sección.

Demostración. Sea $\{\mathbf{b}_k\}_{k=0}^{N-1}$, con $\mathbf{b}_k = (\delta_{(j-k) \bmod N})_{j=0}^{N-1}$, la base canónica de $\mathbb{C}^N$. Como

$$\mathbf{V}_N^k = (\delta_{(i-j-k) \bmod N})_{i,j=0}^{N-1},$$

se puede comprobar fácilmente que $\mathbf{b}_k = \mathbf{V}_N^k\mathbf{b}_0$. Sea $\mathbf{a} \in \mathbb{C}^N$ arbitrario. Entonces, existen únicos $a_0, a_1, \ldots, a_{N-1} \in \mathbb{C}$ de manera que

$$\mathbf{a} = \sum_{k=0}^{N-1} a_k\mathbf{b}_k = \sum_{k=0}^{N-1} a_k\mathbf{V}_N^k\mathbf{b}_0.$$

Si hacemos actuar $\mathbf{H}_N$ sobre $\mathbf{a}$, obtenemos que

$$\mathbf{H}_N\mathbf{a} = \sum_{k=0}^{N-1} a_k(\mathbf{H}_N\mathbf{V}_N^k)\mathbf{b}_0 = \sum_{k=0}^{N-1} a_k(\mathbf{V}_N^k\mathbf{H}_N)\mathbf{b}_0 = \sum_{k=0}^{N-1} a_k\mathbf{V}_N^k\mathbf{h}. \tag{2.12}$$

Teniendo en cuenta que si $\mathbf{h} = (h_0, h_1, \ldots, h_{N-1})$ entonces

$$\mathbf{V}_N^k\mathbf{h} = (h_{(j-k) \bmod N})_{j=0}^{N-1} = (h_{N-k}, h_{N-k+1}, \ldots, h_{N-1}, h_0, \ldots, h_{N-k-1}),$$

podemos expresar (2.12) en forma matricial y obtenemos que

$$\mathbf{H}_N\mathbf{a} = \begin{bmatrix} h_0 & h_{N-1} & \ldots & h_1 \\ h_1 & h_0 & \ldots & h_2 \\ \vdots & \vdots & \ddots & \vdots \\ h_{N-1} & h_{N-2} & \ldots & h_0 \end{bmatrix} \begin{bmatrix} a_0 \\ a_1 \\ \vdots \\ a_{N-1} \end{bmatrix}.$$

En definitiva,

$$\mathbf{H}_N\mathbf{a} = \Big(\sum_{k=0}^{N-1} a_k h_{(n-k) \text{ mód } N}\Big)_{n=0}^{N-1} = \mathbf{a} * \mathbf{h}. \qquad \square$$

Sea $\{\mathbf{b}_k\}_{k=0}^{N-1}$, con $\mathbf{b}_k = (\delta_{(j-k) \text{ mód } N})_{j=0}^{N-1}$ para todo $k = 0, 1, \ldots, N-1$, la base canónica de $\mathbb{C}^N$ y sean $\mathbf{e}_k = (\omega_N^{jk})_{j=0}^{N-1}$ los vectores introducidos en el Teorema (2.1), que coinciden con las columnas de $\mathbf{F}_N$. Para cada $k = 0, 1, \ldots, N-1$, se obtienen las siguientes relaciones:

$$\begin{aligned}
&\mathbf{F}_N\mathbf{b}_k = \mathbf{e}_k,\\
&\mathbf{F}_N\mathbf{e}_k = \mathbf{F}_N^2\mathbf{b}_k = N\mathbf{J}'_N\mathbf{b}_k = N\mathbf{b}_{(-k) \text{ mód } N}.
\end{aligned}$$

Además, puede comprobarse que

$$\mathbf{F}_N\mathbf{V}_N\mathbf{b}_k = \mathbf{F}_N\mathbf{b}_{(k+1) \text{ mód } N} = \mathbf{e}_{(k+1) \text{ mód } N} = \mathbf{M}_N\mathbf{F}_N\mathbf{b}_k, \tag{2.13}$$

donde $\mathbf{M}_N = \operatorname{diag}(\mathbf{e}_1)$ es una matriz de modulación (o desplazamiento frecuencial), que lo que hace es generar un desplazamiento frecuencial [1]. Como consecuencia de (2.13), tenemos que

$$\mathbf{F}_N\mathbf{V}_N = \mathbf{M}_N\mathbf{F}_N, \tag{2.14}$$

luego también, para cada $k = 0, 1, \ldots, N-1$,

$$\mathbf{F}_N\mathbf{V}_N^k = \mathbf{M}_N^k\mathbf{F}_N.$$

Si transponemos todos los términos de la expresión (2.14), obtenemos que

$$\mathbf{V}_N^{\intercal}\mathbf{F}_N = \mathbf{V}_N^{-1}\mathbf{F}_N = \mathbf{F}_N\mathbf{M}_N,$$

de donde se sigue fácilmente que

$$\mathbf{V}_N\mathbf{F}_N = \mathbf{F}_N\mathbf{M}_N^{-1}. \tag{2.15}$$

A continuación vamos a exponer algunas de las propiedades más importantes de la DFT.

Proposición 2.6 (linealidad). *Para cualesquiera* $\mathbf{a}, \mathbf{b} \in \mathbb{C}^N$ *y para todo* $\alpha \in \mathbb{C}$ *se cumple*

I. $\widehat{(\mathbf{a}+\mathbf{b})} = \hat{\mathbf{a}} + \hat{\mathbf{b}}$.

II. $\widehat{(\alpha\mathbf{a})} = \alpha\hat{\mathbf{a}}$.

[1] La multiplicación de la matriz de modulación por un vector en el dominio frecuencial es una operación equivalente a la multiplicación de la matriz de desplazamiento positivo por un vector en el dominio temporal, en el sentido de que la DFT del vector resultante de la primera operación es igual al vector resultante de la segunda operación, como probaremos en la Proposición 2.8. De ahí el nombre de matriz de modulación o de desplazamiento frecuencial.

Demostración. Bastaría con usar la linealidad de $\mathbf{F}$:

I. $\widehat{(\mathbf{a}+\mathbf{b})} = \mathbf{F}(\mathbf{a}+\mathbf{b}) = \mathbf{F}\mathbf{a} + \mathbf{F}\mathbf{b} = \hat{\mathbf{a}} + \hat{\mathbf{b}}$.

II. $\widehat{(\alpha\mathbf{a})} = \mathbf{F}(\alpha\mathbf{a}) = \alpha\mathbf{F}\mathbf{a} = \alpha\hat{\mathbf{a}}$. □

Hemos visto que la matriz de Fourier es invertible, luego el siguiente resultado es directo:

Proposición 2.7. *Para todo $\mathbf{a} \in \mathbb{C}^N$ tenemos que*

$$\mathbf{a} = \mathbf{F}_N^{-1}\hat{\mathbf{a}} = \frac{1}{N}\bar{\mathbf{F}}_N\hat{\mathbf{a}} = \frac{1}{N}\mathbf{J}'_N\mathbf{F}_N\hat{\mathbf{a}}.$$

Proposición 2.8. *Para todo $\mathbf{a} \in \mathbb{C}^N$, se cumple que*

I. $\widehat{(\mathbf{V}_N\mathbf{a})} = \mathbf{M}_N\hat{\mathbf{a}}$.

II. $\widehat{(\mathbf{M}_N^{-1}\mathbf{a})} = \mathbf{V}_N\hat{\mathbf{a}}$.

Demostración. Se sigue directamente de (2.14) y (2.15):

I. $\widehat{(\mathbf{V}_N\mathbf{a})} = \mathbf{F}_N(\mathbf{V}_N\mathbf{a}) = \mathbf{M}_N\mathbf{F}_N\mathbf{a} = \mathbf{M}_N\hat{\mathbf{a}}$.

II. $\widehat{(\mathbf{M}_N^{-1}\mathbf{a})} = \mathbf{F}_N(\mathbf{M}_N^{-1}\mathbf{a}) = \mathbf{V}_N\mathbf{F}_N\mathbf{a} = \mathbf{V}_N\hat{\mathbf{a}}$. □

Teorema 2.9 (convolución cíclica). *Para todo $\mathbf{a}, \mathbf{b} \in \mathbb{C}^N$ se tiene que*

I. $\widehat{(\mathbf{a} * \mathbf{b})} = \hat{\mathbf{a}} \odot \hat{\mathbf{b}}$.

II. $N\widehat{(\mathbf{a} \odot \mathbf{b})} = \hat{\mathbf{a}} * \hat{\mathbf{b}}$,

donde $\mathbf{a} \odot \mathbf{b} = (a_k b_k)_{k=0}^{N-1}$ denota el producto de vectores componente por componente, siendo como de costumbre $\mathbf{a} = (a_k)_{k=0}^{N-1}$ y $\mathbf{b} = (b_k)_{k=0}^{N-1}$.

Demostración. La convolución de los vectores $\mathbf{a}$ y $\mathbf{b}$ es un nuevo vector $\mathbf{c} = (c_j)_{j=0}^{N-1}$ de manera que, para todo $j = 0, 1, \ldots, N-1$,

$$c_j = \sum_{n=0}^{N-1} a_n b_{(j-n) \bmod N}.$$

La DFT de $\mathbf{c}$ es otro vector $\hat{\mathbf{c}} = (\hat{c}_k)_{k=0}^{N-1}$. Para todo $k = 0, 1, \ldots, N-1$ tenemos que

$$\begin{aligned}
\hat{c}_k &= \sum_{j=0}^{N-1} c_j \omega_N^{jk} \\
&= \sum_{j=0}^{N-1} \Big(\sum_{n=0}^{N-1} a_n b_{(j-n) \bmod N} \Big) \omega_N^{jk} \\
&= \sum_{n=0}^{N-1} \Big(\sum_{j=0}^{N-1} a_n b_{(j-n) \bmod N} \Big) \omega_N^{jk} \\
&= \sum_{n=0}^{N-1} a_n \omega_N^{nk} \Big(\sum_{j=0}^{N-1} b_{(j-n) \bmod N} \omega_N^{(j-n)k} \Big).
\end{aligned}$$

Por (2.5), se tiene que $\omega_N^{(j-n)k} = \omega_N^{(j-n)k \bmod N}$, luego

$$\begin{aligned}
\hat{c}_k &= \sum_{n=0}^{N-1} a_n \, \omega_N^{nk} \left(\sum_{j=0}^{N-1} b_{(j-n) \bmod N} \, \omega_N^{(j-n)k \bmod N} \right) \\
&= \sum_{n=0}^{N-1} a_n \, \omega_N^{nk} \left(\sum_{j=0}^{n-1} b_{(j-n) \bmod N} \, \omega_N^{(j-n)k \bmod N} + \sum_{j=n}^{N-1} b_{(j-n) \bmod N} \, \omega_N^{(j-n)k \bmod N} \right) \\
&= \sum_{n=0}^{N-1} a_n \, \omega_N^{nk} \left(\sum_{j=0}^{n-1} b_{N+j-n} \, \omega_N^{(j-n)k \bmod N} + \sum_{j=n}^{N-1} b_{j-n} \, \omega_N^{(j-n)k \bmod N} \right).
\end{aligned}$$

Haciendo el cambio $N + j - n = \ell$, se tiene que

$$\hat{c}_k = \sum_{n=0}^{N-1} a_n \omega_N^{nk} \sum_{\ell=0}^{N-1} b_\ell \, \omega_N^{\ell k} = \hat{a}_k \hat{b}_k,$$

quedando probado el primer apartado.

Sea ahora $\mathbf{c} = \mathbf{a} \odot \mathbf{b} = (a_j b_j)_{j=0}^{N-1}$. En virtud de la Proposición 2.7, tenemos que, para todo $j = 0, 1, \ldots, N-1$,

$$\begin{aligned}
a_j &= \frac{1}{N} \sum_{k=0}^{N-1} \hat{a}_k \omega_N^{-jk}, \\
b_j &= \frac{1}{N} \sum_{\ell=0}^{N-1} \hat{b}_\ell \omega_N^{-j\ell}.
\end{aligned}$$

Entonces, para todo $j = 0, 1, \ldots, N-1$,

$$c_j = a_j b_j = \frac{1}{N^2} \left(\sum_{k=0}^{N-1} \hat{a}_k \omega_N^{-jk} \right) \left(\sum_{\ell=0}^{N-1} \hat{b}_\ell \omega_N^{-\ell k} \right) = \frac{1}{N^2} \sum_{k=0}^{N-1} \sum_{\ell=0}^{N-1} \hat{a}_k \hat{b}_\ell \, \omega_N^{-j(k+\ell)}.$$

Si hacemos $n = k + \ell$ y tenemos en cuenta de nuevo (2.5), obtenemos que

$$c_j = \frac{1}{N^2}\sum_{n=0}^{N-1}\Big(\sum_{k=0}^{N-1}\hat{a}_k\hat{b}_{(n-k) \text{ mód } N}\Big)\omega_N^{-jn} = \frac{1}{N^2}\sum_{n=0}^{N-1}\Big(\hat{a}_n * \hat{b}_n\Big)\omega_N^{-jn}.$$

Es decir, $\mathbf{c} = \frac{1}{N}\mathbf{F}_N^{-1}(\hat{\mathbf{a}} * \hat{\mathbf{b}})$, de donde se sigue que $N\hat{\mathbf{c}} = \hat{\mathbf{a}} * \hat{\mathbf{b}}$ y el segundo apartado queda probado. □

Teorema 2.10 (**identidad de Parseval**). *Para cualesquiera vectores* $\mathbf{a}, \mathbf{b} \in \mathbb{C}^N$*, se cumple que*

$$\frac{1}{N}\langle\hat{\mathbf{a}}, \hat{\mathbf{b}}\rangle = \langle\mathbf{a}, \mathbf{b}\rangle.$$

En particular, tenemos que

$$\frac{1}{N}\|\hat{\mathbf{a}}\|_2^2 = \|\mathbf{a}\|_2^2.$$

Demostración. Dados $\mathbf{a}, \mathbf{b} \in \mathbb{C}^N$, se tiene lo siguiente:

$$\langle\hat{\mathbf{a}}, \hat{\mathbf{b}}\rangle = \langle\mathbf{F}_N\mathbf{a}, \mathbf{F}_N\mathbf{b}\rangle = \mathbf{a}^\intercal\,\mathbf{F}_N\overline{\mathbf{F}}_N\bar{\mathbf{b}} = N\mathbf{a}^\intercal\bar{\mathbf{b}} = N\langle\mathbf{a}, \mathbf{b}\rangle.$$ □

En teoría de la señal, dada una señal finita $\mathbf{f}$, $\|\mathbf{f}\|$ representa su energía. De esta manera, la identidad de Parseval muestra que es equivalente medir la energía de la señal en el dominio temporal o en el frecuencial, en el sentido de que ambas medidas son proporcionales.

2.5. Relación entre la DFT y los coeficientes de Fourier de una función periódica continua

Vamos a analizar cómo podríamos aproximar numéricamente los coeficientes de Fourier c_k de una función $f \in C(\mathbb{T})$ de la que conocemos sus valores en la malla $\{\frac{2\pi j}{N} : j = 0, 1, \ldots, N-1\}$. Supongamos que N es par. Teniendo en cuenta que $f(0) = f(2\pi)$ y la regla del trapecio, tenemos que

$$\begin{aligned}
c_k &= \frac{1}{2\pi}\int_0^{2\pi} f(t)e^{-ikt}dt \\
&\approx \frac{1}{2N}\sum_{j=0}^{N-1}\Big(f\Big(\frac{2\pi j}{N}\Big)e^{-i2\pi jk/N} + f\Big(\frac{2\pi(j+1)}{N}\Big)e^{-i2\pi(j+1)k/N}\Big) \\
&= \frac{1}{2N}\sum_{j=0}^{N-1} f\Big(\frac{2\pi j}{N}\Big)e^{-i2\pi jk/N} + \frac{1}{2N}\sum_{j=1}^{N} f\Big(\frac{2\pi j}{N}\Big)e^{-i2\pi jk/N} \\
&= \frac{1}{N}\sum_{j=0}^{N-1} f\Big(\frac{2\pi j}{N}\Big)e^{-i2\pi jk/N}\ ,
\end{aligned}$$

para cada $k \in \mathbb{Z}$.

Así, para cada $k = 0, 1, \ldots, N-1$, podemos aproximar el k-ésimo coeficiente de Fourier por

$$c_k \approx \hat{f}_k = \frac{1}{N} \sum_{j=0}^{N-1} f\Big(\frac{2\pi j}{N}\Big) \omega_N^{jk} = \frac{1}{N} \hat{\mathbf{f}}[k], \tag{2.16}$$

siendo $\mathbf{f}$ la señal finita dada por $\mathbf{f}[n] = f(\frac{2\pi n}{N})$. Por 2.2, deducimos que los valores $\hat{\mathbf{f}}[k]$ son $N-$periódicos. Como consecuencia de la desigualdad de Bessel (ver [1]), tenemos que $c_k \to 0$ cuando $|k| \to \infty$, por lo que la expresión dada en (2.16) solo tendrá validez para $|k|$ suficientemente pequeño. A continuación, presentamos un ejemplo extraído de [6].

Ejemplo 2.1 Sea f la extensión 2π-periódica de la función

$$f(x) = \begin{cases} 1 & \text{si } x \in (-\frac{\pi}{2}, \frac{\pi}{2}) \\ \frac{1}{2} & \text{si } x \in \{-\frac{\pi}{2}, \frac{\pi}{2}\} \\ 0 & \text{si } x \in [-\pi, -\frac{\pi}{2}) \cup (\frac{\pi}{2}, \pi). \end{cases}$$

cuya gráfica se ilustra en la Figura 2.1.

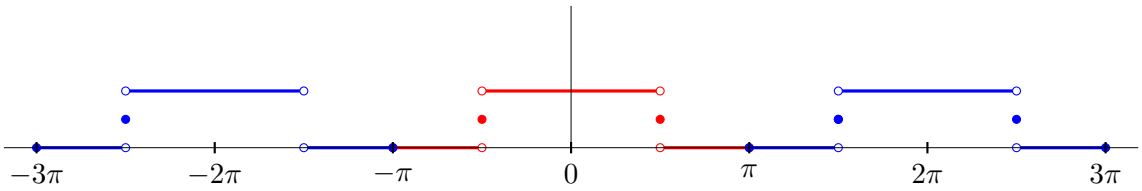


Figura 2.1: Extensión 2π-periódica de la función f.

Notamos que f es par. Entonces, sus coeficientes de Fourier para $k \in \mathbb{Z} \setminus \{0\}$ son:

$$\begin{aligned} c_k &= \frac{1}{2\pi} \int_{-\pi}^{\pi} f(x) e^{-ikx}\, dx \\ &= \frac{1}{2\pi} \int_{-\pi/2}^{\pi/2} \cos kx\, dx - \frac{i}{2\pi} \int_{-\pi/2}^{\pi/2} \operatorname{sen} kx\, dx \\ &= \frac{1}{\pi} \int_{0}^{\pi/2} \cos kx\, dx \\ &= \frac{1}{\pi k} \operatorname{sen} \frac{\pi k}{2}. \end{aligned}$$

Por otro lado,

$$c_0 = \frac{1}{2\pi} \int_{-\pi}^{\pi} f(x) dx = \frac{1}{2\pi} \int_{-\pi/2}^{\pi/2} dx = \frac{1}{2}.$$

Fijado ahora[2] $N \in 4\mathbb{N}$, obtengamos las aproximaciones que nos proporciona la fórmula (2.16):

$$\hat{f}_k = \frac{1}{N}\sum_{j=-N/2}^{N/2-1} f\Big(\frac{2\pi j}{N}\Big)\omega_N^{jk} = \frac{1}{N}\sum_{j=-N/2}^{N/2-1} f\Big(\frac{2\pi j}{N}\Big)\Big(e^{-2\pi i/N}\Big)^{jk}.$$

Ahora, notamos que:

$$f\Big(\frac{2\pi j}{N}\Big) = \begin{cases} 0 & \text{si } j \in \{-\frac{N}{2}, -\frac{N}{2}+1, \ldots, -\frac{N}{4}-1\,,\ \frac{N}{4}+1, \ldots, \frac{N}{2}-1\}. \\ \frac{1}{2} & \text{si } j \in \{-\frac{N}{4}, \frac{N}{4}\}. \\ 1 & \text{si } j \in \{-\frac{N}{4}+1, \ldots, -1,\ 0,\ 1,\ 2, \ldots, \frac{N}{4}-1\}. \end{cases}$$

En base a lo anterior,

$$\begin{aligned} \hat{f}_k &= \frac{1}{N}\Big(1 + \frac{e^{i\pi k/2} + e^{-i\pi k/2}}{2} + \sum_{j=1}^{N/4-1}\big(e^{2\pi ijk/N} + e^{-2\pi ijk/N}\big)\Big) \\ &= \frac{1}{N}\Big(\cos\frac{\pi k}{2} + 1 + 2\sum_{j=1}^{N/4-1}\cos\frac{2\pi jk}{N}\Big). \end{aligned}$$

Así, si $k = \alpha N$, para algún $\alpha \in \mathbb{Z}$:

$$\hat{f}_k = \frac{1}{N}\Big(2 + 2\sum_{j=1}^{N/4-1}\cos 2\pi\alpha j\Big) = \frac{1}{N}\Big(2 + 2\big(\frac{N}{4} - 1\big)\Big) = \frac{1}{2}.$$

Para estudiar el caso $k \in \mathbb{Z}\backslash(N\mathbb{Z})$, haremos uso del núcleo de Dirichlet [6], definido por:

$$D_n(x) = \sum_{k=-n}^{n} e^{ikx},$$

para $x \in \mathbb{R}$, $n \in \mathbb{N}$. Se puede probar que

$$D_n(x) = 1 + 2\sum_{k=1}^{n}\cos(kx),$$

y también que

$$D_n(x) = \frac{\operatorname{sen}\frac{(2n+1)x}{2}}{\operatorname{sen}\frac{x}{2}}.$$

[2] Usaremos la notación $p\mathbb{N}$, con $p \in \mathbb{N}$, para denotar el conjunto de los naturales múltiplos de p.

Usando estas fórmulas, tenemos que:

$$\begin{aligned}
\hat{f}_k &= \frac{1}{N}\left(\cos\frac{\pi k}{2} + 1 + 2\sum_{j=1}^{N/4-1}\cos\left(\frac{2\pi jk}{N}\right)\right)\\
&= \frac{1}{N}\left(\cos\frac{\pi k}{2} + D_{N/4-1}\left(\frac{2\pi k}{N}\right)\right)\\
&= \frac{1}{N}\left(\cos\frac{\pi k}{2} + \frac{\operatorname{sen}\left(\frac{\left(2(\frac{N}{4}-1)+1\right)\frac{2\pi k}{N}}{2}\right)}{\operatorname{sen}\left(\frac{\frac{2\pi k}{N}}{2}\right)}\right)\\
&= \frac{1}{N}\left(\cos\frac{\pi k}{2} + \frac{\operatorname{sen}(\frac{N}{2}-1)\frac{\pi k}{N}}{\operatorname{sen}\frac{\pi k}{N}}\right)\\
&= \frac{1}{N}\left(\cos\frac{\pi k}{2} + \frac{\operatorname{sen}(\frac{\pi k}{2}-\frac{\pi k}{N})}{\operatorname{sen}\frac{\pi k}{N}}\right)\\
&= \frac{1}{N}\left(\cos\frac{\pi k}{2} + \frac{\operatorname{sen}\frac{\pi k}{2}\cos\frac{\pi k}{N} - \operatorname{sen}\frac{\pi k}{N}\cos\frac{\pi k}{2}}{\operatorname{sen}\frac{\pi k}{N}}\right)\\
&= \frac{1}{N}\operatorname{sen}\frac{\pi k}{2}\operatorname{cotg}\frac{\pi k}{N}.
\end{aligned}$$

Este ejemplo deja ver el diferente comportamiento asintótico de los coeficientes de Fourier c_k y sus aproximaciones $\hat{f}_k$ al hacer $|k| \to \infty$ (ver Figura 2.2).

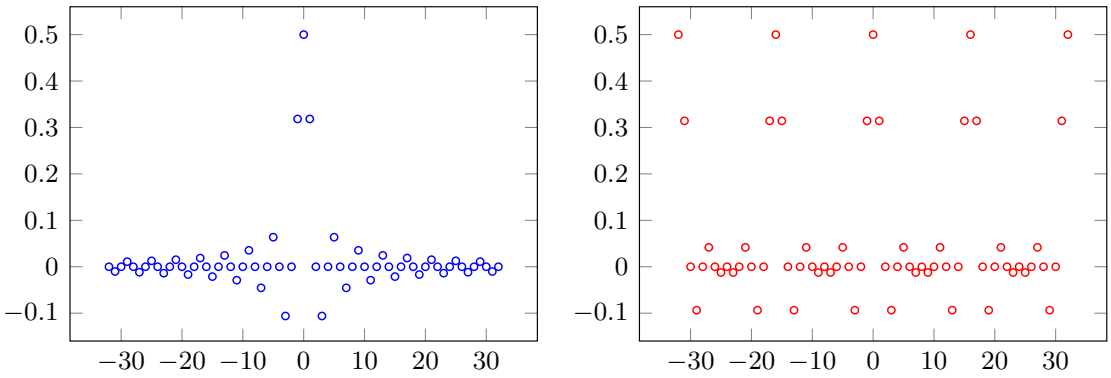


Figura 2.2: A la izquierda, los coeficientes de Fourier c_k y a la derecha sus aproximaciones $\hat{f}_k$.

A continuación daremos un teorema que será útil después para acotar el error de aproximación.

Teorema 2.11. *Sea $f \in \mathbb{C}(\mathbb{T})$ tal que $\sum_{k\in\mathbb{Z}} |c_k| < \infty$. Entonces, para cualquier $k \in \mathbb{Z}$, se cumple la siguiente identidad:*

$$\hat{f}_k = \sum_{\ell\in\mathbb{Z}} c_{k+\ell N}. \tag{2.17}$$

Demostración. Como $\sum_{k\in\mathbb{Z}} |c_k| < \infty$, la serie converge. Si $x \in \mathbb{T}$, entonces

$$f(x) = \sum_{\ell\in\mathbb{Z}} c_\ell e^{i\ell x}. \tag{2.18}$$

Por otro lado,

$$\hat{f}_k = \frac{1}{N}\sum_{j=0}^{N-1} f(\frac{2\pi j}{N})\,\omega_N^{jk} = \frac{1}{N}\sum_{j=0}^{N-1}\sum_{\ell\in\mathbb{Z}} c_\ell e^{2\pi i j\ell/N}\omega_N^{jk}$$

Por 2.1 y usando que la serie (2.18) es uniformemente convergente, pudiendo consecuentemente intercambiar los sumatorios, se cumple

$$\hat{f}_k = \frac{1}{N}\sum_{\ell\in\mathbb{Z}} c_\ell \sum_{j=0}^{N-1} \omega_N^{-jl}\omega_N^{jk} = \frac{1}{N}\sum_{\ell\in\mathbb{Z}} c_\ell \sum_{j=0}^{N-1} \omega_N^{j(k-\ell)}.$$

Aplicando el Lema 2.3, concluimos finalmente que

$$\hat{f}_k = \sum_{\ell\in\mathbb{Z}} c_\ell \delta_{(k-\ell) \text{ mód } N} = \sum_{\ell\in\mathbb{Z}} c_{k+\ell N}. \qquad \square$$

Corolario 2.12. *Bajo las condiciones del Teorema 2.11, tenemos la siguiente cota para el error*

$$|e_k| = |\hat{f}_k - c_k| \le \sum_{\ell\in\mathbb{Z}\setminus 0} |c_{k+\ell N}|$$

para $k = -\frac{N}{2}, -\frac{N}{2}+1, \ldots, \frac{N}{2}-1$. *Además, si* $f \in C^r(\mathbb{T})$ *con* $r \in \mathbb{N}$ *y existe una constante* $c > 0$ *tal que*

$$|c_k| \le \frac{c}{|k|^{r+1}}, \quad k \in \mathbb{Z}\setminus\{0\}, \tag{2.19}$$

entonces, para $|k| < \frac{N}{2}$, *se verifica la siguiente cota del error:*

$$|e_k| = |\hat{f}_k - c_k| \le \frac{c}{rN^{r+1}}\left(\left(\frac{1}{2}+\frac{k}{N}\right)^{-r} + \left(\frac{1}{2}-\frac{k}{N}\right)^{-r}\right). \tag{2.20}$$

Demostración. Usando (2.17), tenemos que

$$\hat{f}_k = \sum_{\ell\in\mathbb{Z}} c_{k+\ell N} = c_k + \sum_{\ell\in\mathbb{Z}\setminus\{0\}} c_{k+\ell N}.$$

Así,

$$|\hat{f}_k - c_k| \le \sum_{\ell=1}^{\infty}(|c_{k+\ell N}| + |c_{k-\ell N}|).$$

Ahora, usando la desigualdad (2.19) tenemos que

$$|\hat{f}_k - c_k| \leq \frac{c}{N^{r+1}} \sum_{\ell=1}^{\infty} \left(\left| \frac{k}{N} + \ell \right|^{-r-1} + \left| \frac{k}{N} - \ell \right|^{-r-1} \right)$$
$$= \frac{c}{N^{r+1}} \sum_{\ell=1}^{\infty} \left((\ell + \frac{k}{N})^{-r-1} + (\ell - \frac{k}{N})^{-r-1} \right), \tag{2.21}$$

donde la última igualdad se debe a que $|k| < \frac{N}{2}$. Si $|s| < \frac{1}{2}$ y $\ell \in \mathbb{N}$ se puede comprobar que

$$(\ell + s)^{-r-1} < \int_{\ell-1/2}^{\ell+1/2} (x + s)^{-r-1} dx,$$

ya que la función $g(x) = (x + s)^{-r-1}$ es convexa y monótona decreciente. Por tanto,

$$\sum_{\ell=1}^{\infty} (\ell + s)^{-r-1} < \int_{1/2}^{\infty} (x + s)^{-r-1} dx = \frac{1}{r} (\frac{1}{2} + s)^{-r}. \tag{2.22}$$

Como $|\frac{k}{N}| < \frac{1}{2}$, se puede aplicar la acotación (2.22) a la expresión (2.21), y así se llega a (2.20). □

En la Tabla 2.1 ilustramos el comportamiento del error de aproximación de algunos de los coeficientes del Ejemplo 2.1. Por simetría, solo se muestran los resultados para índices positivos. Cuando el índice es par, ambos valores son cero, por lo que omitimos también los resultados para índices pares. Vemos que los resultados son coherentes con la cota 2.20, siendo las aproximaciones más exactas en los resultados con índices centrales y menos precisas en los extremos.

k	c_k	$\hat{f}_k$	e_k
0	0.500000	0.50000000	0.00000000
1	0.318310	0.31420872	$4.10116793 \cdot 10^{-3}$
3	−0.106103	−0.09353786	$1.25654352 \cdot 10^{-2}$
5	0.063662	0.04176116	$2.19008124 \cdot 10^{-2}$
7	−0.045473	−0.01243202	$3.30408179 \cdot 10^{-2}$
9	0.035368	−0.01243202	$4.77997881 \cdot 10^{-2}$
11	−0.028937	0.04176116	$7.06984273 \cdot 10^{-2}$
13	0.024485	−0.09353786	$1.18023236 \cdot 10^{-1}$
15	−0.021221	0.31420872	$3.35429377 \cdot 10^{-1}$
17	0.018724	0.31420872	$2.95484607 \cdot 10^{-1}$

Tabla 2.1: Comparación de coeficientes de Fourier y sus aproximaciones del Ejemplo 2.1.

2.6. Aproximación numérica de la transformada de Fourier continua

La transformada discreta también será muy útil para aproximar numéricamente la transformada de Fourier de una función $f \in L^1(\mathbb{R}) \cap C(\mathbb{R})$. Para $n \in \mathbb{N}$ suficientemente grande que, sin pérdida de generalidad, supondremos par y para cualquier $\omega \in \mathbb{R}$ tenemos que

$$\hat{f}(\omega) = \int_{\mathbb{R}} f(t)\, e^{-it\omega}\, dt \approx \int_{-n\pi}^{n\pi} f(x)\, e^{-it\omega}\, dt. \tag{2.23}$$

Consideramos la colección de puntos $\{\frac{2\pi j}{N} : j = -\frac{nN}{2}, \ldots, \frac{nN}{2} - 1\}$ del intervalo $[-n\pi, n\pi)$ y aproximamos la integral de (2.23) mediante la fórmula del rectángulo:

$$\int_{-n\pi}^{n\pi} f(t)\, e^{-it\omega} dx \approx \frac{2\pi}{N} \sum_{j=-nN/2}^{nN/2-1} f\Big(\frac{2\pi j}{N}\Big)\, e^{-2\pi i j\omega/N}.$$

Tomando $\omega = \frac{k}{N}$ para $k = -\frac{nN}{2}, \ldots, \frac{nN}{2} - 1$, obtenemos el siguiente valor aproximado de $\hat{f}(\frac{k}{N})$:

$$\hat{f}\Big(\frac{k}{N}\Big) \approx \frac{2\pi}{N} \sum_{j=-nN/2}^{nN/2-1} f\Big(\frac{2\pi j}{N}\Big)\, e^{-2\pi i j k/nN}.$$

Capítulo 3

Cálculo óptimo de la DFT. La transformada rápida de Fourier

En este capítulo vamos a centrarnos en diseñar algoritmos para calcular la DFT de manera eficiente. En particular, nos centraremos en el algoritmo de la Transformada Rápida de Fourier o FFT (del inglés, Fast Fourier Transform). Ya mencionamos en el capítulo anterior que el cálculo directo de la DFT de un vector de tamaño N tiene un coste del orden de $O(N^2)$ operaciones. Un algoritmo de transformada rápida de Fourier reduce el número de operaciones hasta el orden de $O(N \log N)$.

La historia de la Transformada Rápida de Fourier (FFT) tiene una característica recurrente: la omnipresencia de Carl Friedrich Gauss(1777-1855). Ya sea que esté resolviendo ecuaciones, calculando trayectorias de cuerpos celestes o simplemente tomando un café en la cafetería de la Universidad de Göttingen, Gauss siempre se asoma. Entonces, ¿quién se sorprende de que también encontremos sus huellas en la FFT? Aunque Gauss no describió un algoritmo de FFT como tal, su trabajo a principios del siglo XIX en el cálculo de la órbita del asteroide Pallas ofrece un interesante precursor de los conceptos que eventualmente conducirían a su desarrollo.

En su intento por determinar la órbita del asteroide, Gauss se encontró con la necesidad de ajustar 12 datos de posición utilizando un polinomio trigonométrico. Este problema de interpolación esencialmente requería calcular una DFT de tamaño 12.

Para reducir el número de operaciones necesarias, Gauss aplicó un ingenioso enfoque. Descompuso la DFT de tamaño 12 en DFT más pequeñas de tamaños 3 y 4. Esta estrategia de descomposición es similar a la idea subyacente en la FFT moderna, donde una DFT de tamaño grande se descompone en varias DFT más pequeñas cuyos tamaños son factores primos, lo que resulta en una reducción significativa del número total de operaciones requeridas.

Así, aunque Gauss no llegó a formalizar un algoritmo de FFT, su trabajo precursor en la descomposición de la DFT sienta las bases para su desarrollo posterior y demuestra su genialidad

matemática incluso en campos aparentemente alejados de la ciencia de la computación.

La creación de la FFT se debe a James Cooley y John Tukey, quienes describieron el primer algoritmo de transformada rápida de Fourier en el artículo *An algorithm for the machine calculation of complex Fourier series*, publicado en la revista *Mathematics of Computation* en abril de 1965.

Antes de adentrarnos en la FFT, repasamos algunos conceptos básicos de cálculo numérico. El *coste computacional* de un algoritmo viene determinado por el número de operaciones en punto flotante, es decir, las sumas y multiplicaciones, ya sean reales o complejas, necesarias para llevar a cabo el algoritmo. Normalmente nos interesaremos por el orden de magnitud del coste computacional de un algoritmo y utilizaremos la notación de la *O grande*.

Además del coste computacional, en un algoritmo nos interesa cuantificar el *coste de almacenamiento*, es decir, la cantidad de memoria que se requiere; lo ideal es que sea lo menor posible. Para reducir el tiempo de computación, es de gran interés descomponer el algoritmo en pequeños subalgoritmos independientes que son llevados a cabo simultáneamente. Esto es lo que se entiende por *programación paralela*.

Otro aspecto a tener en cuenta está relacionado con la propagación de los errores: un algoritmo se dice que es *numéricamente estable* si pequeñas perturbaciones en los datos de entrada no suponen un gran cambio en las variables de salida.

3.1. Introducción

Uno de los principales motivos para usar métodos de Fourier es la existencia de algoritmos rápidos para implementar la DFT. Dado $N \in \mathbb{N}$ y un vector $\mathbf{a} = (a_j)_{j=0}^{N-1} \in \mathbb{C}^N$, recordamos que su DFT es otro vector $\hat{\mathbf{a}} = (\hat{a}_k)_{k=0}^{N-1} \in \mathbb{C}^N$ donde, para todo $k = 0, 1, \ldots, N-1$,

$$\hat{a}_k = \sum_{j=0}^{N-1} a_j \omega_N^{jk}, \tag{3.1}$$

siendo $\omega_N = e^{-2\pi i/N}$. Esta es la forma en suma de la DFT (también mencionamos la representación matricial en el capítulo anterior). Aplicando la técnica *divide y vencerás*, la FFT realiza la suma (3.1) mediante la evaluación reiterativa de sumas parciales. Dicha técnica es una herramienta muy útil para reducir el tiempo de ejecución de un algoritmo. El problema original se descompone en varios subproblemas de tamaño más pequeño pero con la misma estructura. La descomposición mencionada se aplica entonces de forma iterativa para reducir aún más los subproblemas. Para aplicar esta técnica a la construcción de un algoritmo FFT se necesita una indexación adecuada.

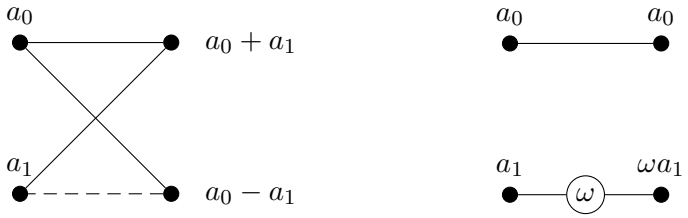
Existe una herramienta muy interesante para mostrar la estructura de un algoritmo y simplificar

su programación. Se trata de los grafos de flujo de señal (SFG, del inglés *signal flow graph*). Un SFG es un grafo cuyos vértices representan los resultados intermedios y sus aristas las operaciones aritméticas. Para nosotros, todos los grafos de flujo de señal tendrán formas que recuerdan a las alas de una mariposa. De hecho también se los conoce con el nombre de grafos mariposa.

Un grafo de flujo de señal siempre se lee de izquierda a derecha. Por ejemplo, en la Figura 3.1 se muestran los SFG de las operaciones **Ma** y **Na** definidas por las matrices

$$\mathbf{M} = \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} \qquad \text{y} \qquad \mathbf{N} = \begin{bmatrix} 1 & 0 \\ 0 & \omega \end{bmatrix},$$

con $\omega \in \mathbb{C}$, y el vector $\mathbf{a} = (a_0, a_1)^\intercal$.



(a) SFG de **Ma** **(b)** SFG de **Na**

Figura 3.1: Grafos de flujo de señal para las operaciones **Ma** y **Na**.

En la Figura 3.1a, el valor numérico correspondiente a un nodo de la derecha depende de los valores en los nodos iniciales de las aristas concurrentes en dicho nodo. Una arista continua indica que el nodo inicial suma y una arista discontinua indica que ese nodo resta. Además, si en el medio de una arista aparece un coeficiente, el valor del nodo inicial debe multiplicarse por dicho coeficiente, como en la Figura 3.1b.

Como ya se mencionó en el capítulo anterior, el mismo algoritmo que calcule la DFT servirá para calcular la DFT inversa, en virtud del Lema 2.4.

3.2. FFT de base 2

Si N es una potencia de 2 y queremos calcular la DFT de un vector de tamaño N, se utilizan los denominados algoritmos FFT de base 2, que se inspiran en la técnica *divide y vencerás*. Dos de los algoritmos FFT de base 2 más habituales son el algoritmo de Sande-Tukey y el de Cooley-Tukey; ambos son iguales en cuanto a coste computacional. Mientras que el algoritmo de Sande-Tukey reduce la cantidad de cálculos de la DFT dividiendo la señal de entrada en subconjuntos en el dominio de la frecuencia (*decimación en frecuencia*), el algoritmo de Cooley-Tukey simplifica los cálculos de la DFT agrupando los datos de entrada en subconjuntos en el dominio del tiempo (*decimación en tiempo*), antes de realizar cualquier operación en el dominio frecuencial.

3.2.1. Algoritmo de Sande-Tukey

Supongamos que $N = 2^t$, con $t \in \mathbb{N}\backslash\{1\}$. Así, dado un vector $\mathbf{a} = (a_n)_{n=0}^{N-1} \in \mathbb{C}^N$, podemos expresar su DFT componente a componente como sigue:

$$\begin{aligned}\hat{a}_k &= \sum_{j=0}^{N/2-1} a_j\,\omega_N^{jk} + \sum_{j=N/2}^{N-1} a_j\,\omega_N^{jk} \\ &= \sum_{j=0}^{N/2-1} a_j\,\omega_N^{jk} + \sum_{j=0}^{N/2-1} a_{N/2+j}\,\omega_N^{(N/2+j)k} \\ &= \sum_{j=0}^{N/2-1} (a_j + (-1)^k a_{N/2+j})\,\omega_N^{jk}, \end{aligned} \tag{3.2}$$

donde para la segunda igualdad hemos usado que $(\omega_N^k)^{N/2} = (-1)^k$. Distingamos ahora entre las componentes con subíndice par e impar. Para cada $k = 0, 1, \ldots, \frac{N}{2} - 1$ obtenemos que

$$\hat{a}_{2k} = \sum_{j=0}^{N/2-1} (a_j + a_{N/2+j})\,\omega_N^{j(2k)} = \sum_{j=0}^{N/2-1} (a_j + a_{N/2+j})\,\omega_{N/2}^{jk}\,, \tag{3.3}$$

$$\hat{a}_{2k+1} = \sum_{j=0}^{N/2-1} (a_j - a_{N/2+j})\,\omega_N^{j(2k+1)} = \sum_{j=0}^{N/2-1} (a_j - a_{N/2+j})\,\omega_N^{j}\,\omega_{N/2}^{jk}. \tag{3.4}$$

Ahora, usamos la siguiente notación:

$$g_j^1 = a_j + a_{N/2+j}, \tag{3.5}$$

$$g_j^2 = (a_j - a_{N/2+j})\,\omega_N^j. \tag{3.6}$$

con $j = 0, 1, \ldots, \frac{N}{2} - 1$. De este modo,

$$\hat{a}_{2k} = \sum_{j=0}^{N/2-1} g_j^1\,\omega_{N/2}^{kj}, \tag{3.7}$$

$$\hat{a}_{2k+1} = \sum_{j=0}^{N/2-1} g_j^2\,\omega_{N/2}^{kj}. \tag{3.8}$$

Notamos que las expresiones (3.3) y (3.4) son, respectivamente, las transformadas discretas de Fourier de los vectores $\mathbf{g}^1 = (g_n^1)_{n=0}^{N/2-1}$ y $\mathbf{g}^2 = (g_n^2)_{n=0}^{N/2-1}$ de $\mathbb{C}^{N/2}$. Tenemos dos opciones, se puede detener este proceso de descomposición y aplicar directamente (3.3) y (3.4), o continuar el proceso de forma recursiva t veces, haciendo que los vectores $\mathbf{g}^1$ y $\mathbf{g}^2$ jueguen el papel del vector $\mathbf{a}$. Este proceso recursivo se denomina algoritmo de Sande-Tukey.

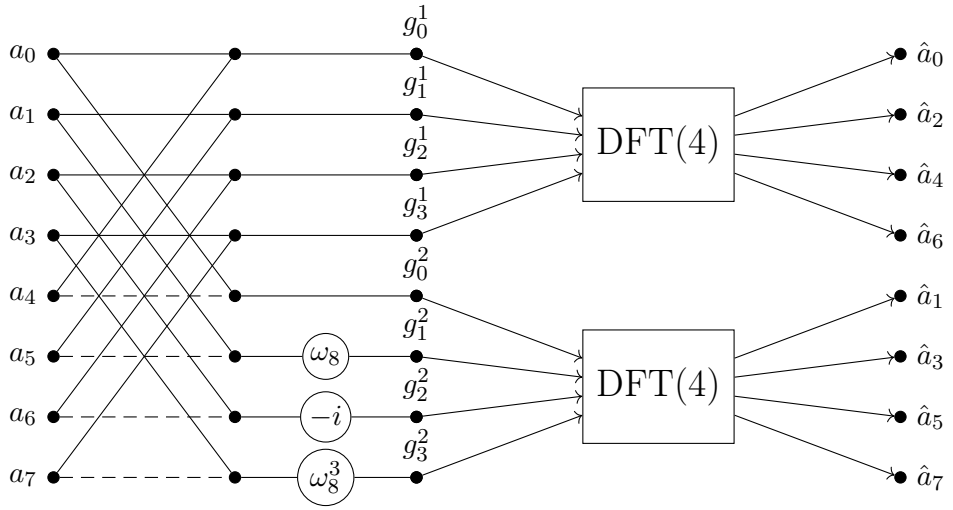


Figura 3.2: Descomposición de una DFT(8) en dos DFT(4)

En la Figura 3.2 se ilustra la primera opción para una DFT(8). Los dos primeros bloques de aristas se corresponden con el cálculo de g_j^1 y g_j^2 para todo $j = 0, 1, 2, 3$, según las fórmulas (3.5) y (3.6). Después, en las dos cajas que siguen se realiza el cálculo de las dos DFT(4) de los vectores $\mathbf{g}^1$ y $\mathbf{g}^2$ según las fórmulas (3.7) y (3.8). Haciendo esto ya se consigue reducir el número de operaciones considerablemente.

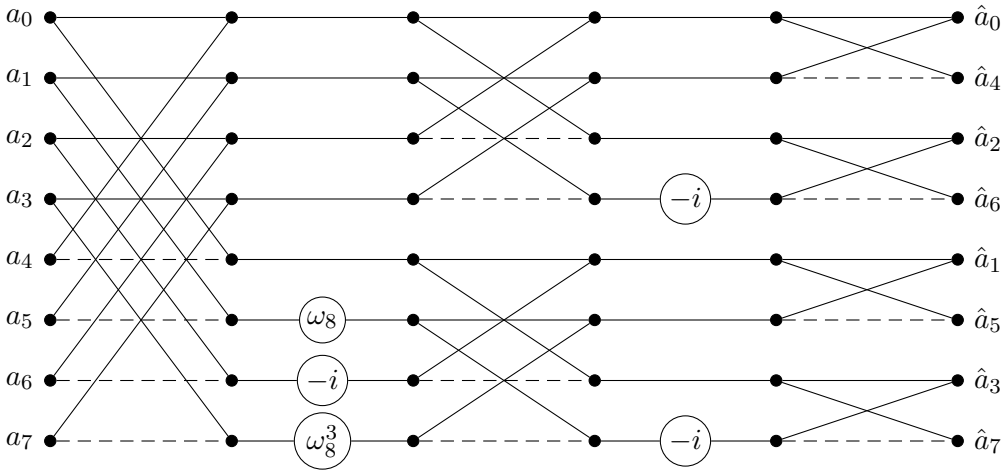


Figura 3.3: Algoritmo de Sande-Tukey para una DFT(8)

Sin embargo, el algoritmo de Sande-Tukey va más allá. En vez de realizar el cálculo directo de las dos DFT(4) de $\mathbf{g}^1$ y $\mathbf{g}^2$, lo que se hace es repetir los pasos anteriores, haciendo que cada una de ellas pase a jugar el papel del vector a inicial. El SFG del algoritmo de Sande-Tukey para una DFT de tamaño 8 se ilustra en la Figura 3.3. En este caso, el proceso se repetiría 3 veces ($8 = 2^3$). Si nos fijamos, nos damos cuenta de por qué resulta interesante ilustrar esta clase de algoritmos mediante grafos de flujo de señal. El dibujo que forman las aristas del primer bloque en la Figura

3.3 se va repitiendo en los sucesivos a escala más pequeña. Esto resulta muy ilustrativo a la hora de programar el algoritmo.

Observamos que los índices de la salida no están en el orden natural. De hecho, el algoritmo reordena los índices según un cierto orden que explicamos a continuación. Cualquiera de los índices $k \in J = \{0, 1, \ldots, N-1\} = \{0, 1, \ldots, 2^t - 1\}$ puede escribirse en forma binaria:

$$k = (k_{t-1}, \ldots, k_1, k_0)_2 = k_{t-1} \cdot 2^{t-1} + \cdots + k_1 \cdot 2 + k_0 \cdot 2^0,$$

con $k_j \in \{0, 1\}$. La permutación $\rho : J \to J$ definida por

$$\rho(k) = (k_0, k_1, \ldots, k_{t-1}) = k_0 \cdot 2^{t-1} + \cdots + k_{t-2} \cdot 2^1 + k_{t-1} \cdot 2^0$$

permite obtener el índice reordenado según el orden de bits inverso.

A partir de la construcción que acabamos de describir, podemos establecer el siguiente resultado, cuya demostración puede consultarse en [6]:

Teorema 3.1. *Sea* $\mathbf{a} = (a_n)_{n=0}^{N-1} \in \mathbb{C}^N$ *un vector dado. El algoritmo de Sande-Tukey calcula la Transformada de Fourier Discreta de* $\mathbf{a}$ *en el orden de bits inverso, es decir, devuelve el vector* $(\hat{a}_{\rho(k)})_{k=0}^{N-1}$.

En el caso de una DFT de tamaño 8, la Tabla 3.1 ilustra cómo reordena los índices el algoritmo de Sande-Tukey.

a_k	$a_{k_2k_1k_0}$	$a_{k_0k_1k_2}$	$\hat{a}_{\rho(k)}$
a_0	a_{000}	a_{000}	$\hat{a}_0$
a_1	a_{001}	a_{100}	$\hat{a}_4$
a_2	a_{010}	a_{010}	$\hat{a}_2$
a_3	a_{011}	a_{110}	$\hat{a}_6$
a_4	a_{100}	a_{001}	$\hat{a}_1$
a_5	a_{101}	a_{101}	$\hat{a}_5$
a_6	a_{110}	a_{011}	$\hat{a}_3$
a_7	a_{111}	a_{111}	$\hat{a}_7$

Tabla 3.1: Reordenación de índices del algoritmo de Sande-Tukey al calcular una DFT(8)

A la hora de implementar el algoritmo en algún software, es necesario reordenar los elementos del vector de entrada o de salida según el orden de bits inverso. En nuestro caso, utilizaremos MATLAB, que ofrece un paquete de herramientas llamado *Signal Processing Toolbox*, diseñado para trabajar con señales. Este paquete incluye una función llamada *bitrevorder*, que reordena

directamente los elementos de un vector de entrada según el orden de bits inverso. Se puede comprobar que, introduciendo los elementos del vector de entrada según el orden de bits inverso, entonces en la salida aparecen en el orden natural, tal y como se muestra en la Figura 3.4 para una DFT(8).

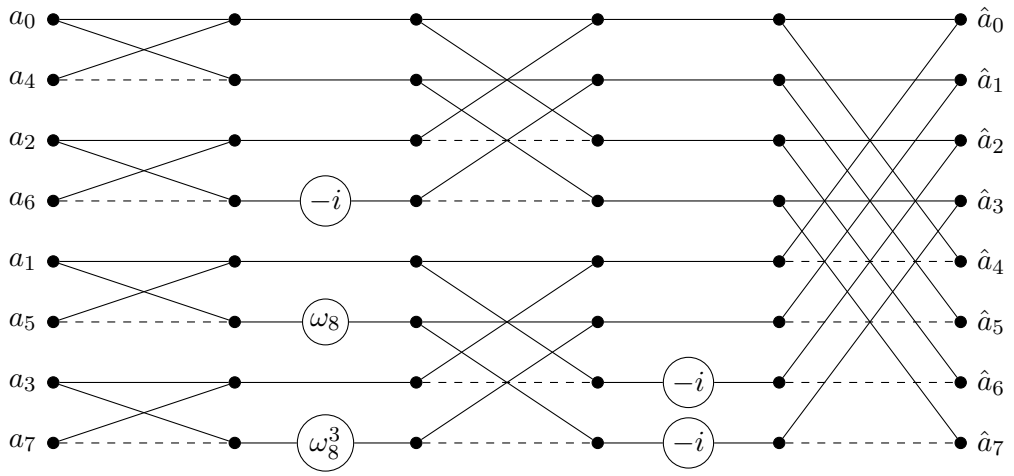


Figura 3.4: Algoritmo de Sande-Tukey para una DFT(8) con los índices del vector de entrada en orden de bits inverso

3.2.2. Algoritmo de Cooley-Tukey

De nuevo, consideramos $N = 2^t$ con $t \in \mathbb{N}\backslash\{1\}$ dado. Sea $\mathbf{a} = (a_n)_{n=0}^{N-1} \in \mathbb{C}^N$ y consideremos las vectores f_n^1 y f_n^2 correspondientes a los elementos de $\mathbf{a}$ con subíndice par e impar respectivamente, es decir, para todo $n = 0, 1, \ldots, N-1$,

$$f_n^1 = a_{2n},$$
$$f_n^2 = a_{2n+1}.$$

Esta descomposición del vector $\mathbf{a}$ en dos vectores de manera que uno contenga aquellos elementos con subíndice par y el otro los de subíndice par, la denominaremos *decimación en el dominio temporal*. Ahora la idea es muy similar a lo que ya describimos para construir el algoritmo anterior:

$$\hat{a}_k = \sum_{j=0}^{N-1} a_j \omega_N^{jk} = \sum_{j=0}^{N/2-1} a_{2j} \omega_N^{2jk} + \sum_{j=0}^{N/2-1} a_{2j+1} \omega_N^{k(2j+1)},$$

para todo $k = 0, 1, \ldots, N-1$. Usando que $\omega_N^2 = \omega_{N/2}$ y las expresiones de f_n^1 y f_n^2, tenemos que

$$\hat{a}_k = \sum_{j=0}^{N/2-1} f_j^1 \omega_{N/2}^{jk} + \sum_{j=0}^{N/2-1} f_j^2 \omega_N^k \omega_{N/2}^{jk} = \hat{f}_k^1 + \omega_N^k \hat{f}_k^2,$$

siendo $\hat{f}_k^1$ y $\hat{f}_k^2$ los elementos k-ésimos de las DFT($N/2$) de los vectores $(f_n^1)_{n=0}^{N-1}$ y $(f_n^2)_{n=0}^{N-1}$ respectivamente. Consideremos las extensiones $(N/2)$-periódicas de los vectores $\hat{f}_k^1$ y $\hat{f}_k^2$. De esta manera, $\hat{f}_k^1 = \hat{f}_{k+N/2}^1$ y $\hat{f}_k^2 = \hat{f}_{k+N/2}^2$. Además, $\omega_N^{k+N/2} = -\omega_N^k$. Por tanto,

$$\begin{aligned}\hat{a}_k &= \hat{f}_k^1 + \omega_N^k \hat{f}_k^2,\\ \hat{a}_{k+N/2} &= \hat{f}_k^1 - \omega_N^k \hat{f}_k^2,\end{aligned}$$

para todo $k = 0, 1, \ldots, N/2 - 1$.

Ahora, notamos que el coste computacional de calcular directamente $\hat{f}_k^1$ es $(N/2)^2$ y lo mismo para $\hat{f}_k^2$. Además, todavía habría que realizar $(N/2)$ multiplicaciones por el factor ω_N^k. Para reducir más el coste computacional, lo que se hace es repetir el proceso de decimación para los vectores $(f_n^1)_{n=0}^{N-1}$ y $(f_n^2)_{n=0}^{N-1}$, haciendo que jueguen el papel de **a** al principio. Es decir, construiríamos dos vectores a partir de $(f_n^1)_{n=0}^{N-1}$, uno que contuviera los elementos con subíndice par y otro los de subíndice impar y de forma análoga para $(f_n^2)_{n=0}^{N-1}$. De esta manera, obtenemos 4 vectores de tamaño $N/4$:

$$\begin{aligned}g_n^{11} &= f_{2n}^1,\\ g_n^{12} &= f_{2n+1}^1,\\ g_n^{21} &= f_{2n}^2,\\ g_n^{22} &= f_{2n+1}^2,\end{aligned}$$

para todo $n = 0, 1, \ldots, N/4 - 1$. A partir de aquí obtendríamos, para cada $k = 0, 1, \ldots, N/2 - 1$, $\hat{f}_k^1$ y $\hat{f}_k^2$ como sigue:

$$\begin{aligned}\hat{f}_k^1 &= \hat{g}_k^{11} + \omega_{N/2}^k \hat{g}_k^{12},\\ \hat{f}_{k+N/4}^1 &= \hat{g}_k^{11} - \omega_{N/2}^k \hat{g}_k^{12},\\ \hat{f}_k^2 &= \hat{g}_k^{21} + \omega_{N/2}^k \hat{g}_k^{22},\\ \hat{f}_{k+N/4}^2 &= \hat{g}_k^{21} - \omega_{N/2}^k \hat{g}_k^{22},\end{aligned}$$

donde $(\hat{g}_k^{ij})_{k=0}^{N/4-1}$ son las DFT($N/4$) de los vectores $(g_n^{ij})_{n=0}^{N/4-1}$. La decimación en tiempo puede repetirse $t = \log_2 N$ veces. Igual que ocurría con Sande-Tukey, el algoritmo tal cual lo hemos descrito da la salida de la DFT con los subíndices del vector según el orden de bits inverso, por lo que a la hora de implementar el algoritmo, es necesario reordenar los índices del vector de entrada o salida según el orden de bits inverso. En la Figura 3.5 puede verse el SFG del algoritmo de Cooley-Tukey para una DFT(8) con la salida del vector con índices en orden natural.

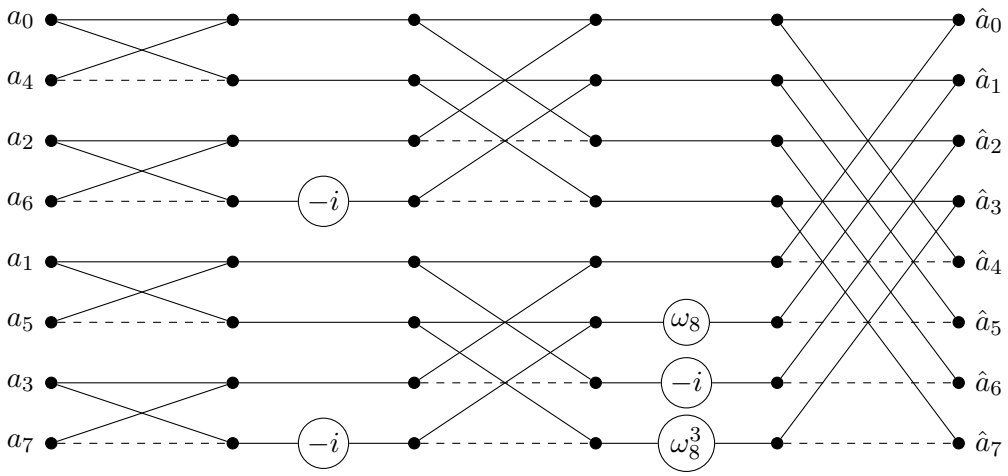


Figura 3.5: Algoritmo de Cooley-Tukey para una DFT(8)

3.2.3. Coste computacional de los FFT de base 2

El cálculo de la DFT requiere realizar multiplicaciones y sumas de números complejos. A su vez, dados dos números complejos $a = \alpha + i\beta$ y $\omega = \omega_0 + i\omega_1$, donde $\alpha, \beta, \omega_0, \omega_1 \in \mathbb{R}\backslash\{0\}$, la multiplicación de a por ω supone cuatro multiplicaciones reales y dos sumas reales. Suponiendo que los valores $\omega_0 \pm \omega_1$ ya están almacenados en el ordenador, tenemos que hacer tres multiplicaciones y tres sumas reales:

$$\Re(a\omega) = (\alpha + \beta)\omega_0 - \alpha(\omega_0 + \omega_1),$$
$$\Im(a\omega) = (\alpha + \beta)\omega_0 - \alpha(\omega_0 - \omega_1),$$

donde $\Re$ y $\Im$ denotan la parte real e imaginaria, respectivamente, de un número complejo. Las multiplicaciones por los factores ± 1 y $\pm i$ son triviales y no las tendremos en cuenta. Por otro lado, la multiplicación por una raíz octava de la unidad, $\omega_8^k = \frac{\pm 1 \pm i}{\sqrt{2}}$ con $k = 0, 1, \ldots, 7$, requiere 2 multiplicaciones y 2 sumas reales, mientras que la multiplicación por una raíz n-ésima de la unidad con $n \in \mathbb{N}\backslash\{1, 2, 4, 8\}$ se puede implementar mediante un algoritmo que supone 3 multiplicaciones y 3 sumas reales. Finalmente, la suma de dos números complejos supone 2 sumas reales.

Denotemos por $\mu(t)$ el número de multiplicaciones reales para el cálculo de una DFT(2^t) mediante un algoritmo FFT de base 2 y por $\alpha(t)$ el número de sumas reales. Veamos cuántas operaciones son necesarias para el cálculo de una DFT de tamaño 8. Para ello, nos fijamos en la Figura 3.3. El número de multiplicaciones reales que se hace es 4, ya que solo se realizan 2 multiplicaciones por raíces octavas de la unidad (recordamos que no contamos multiplicaciones por los factores ± 1 y $\pm i$). Estas dos multiplicaciones por raíces octavas de la unidad suponen a su vez realizar dos sumas reales cada una. A esto, habría que añadirle el cálculo de 8 sumas complejas en cada

etapa. Es decir, 16 sumas reales por etapa y un total de $3 \cdot 16 = 52$. Resumiendo,

$$\begin{aligned} \mu(3) &= 4, \\ \alpha(3) &= 4 + 2 \cdot 3 \cdot 8 = 52. \end{aligned}$$

Si ahora tomamos $N = 2^t$ siendo t un natural distinto de 1, para obtener la DFT($2N$) tenemos que calcular dos DFT de tamaño N, y por lo tanto realizar $2N$ sumas complejas y N multiplicaciones complejas por factores de rotación ω_{2N}^j con $j = 0, 1, \ldots, N-1$. Para cada DFT de tamaño N tendríamos que realizar $\alpha(t)$ sumas y $\mu(t)$ multiplicaciones reales. Por otro lado, las $2N = 2 \cdot 2^t$ sumas complejas suponen el doble de sumas reales, es decir, $2 \cdot 2^{t+1}$. Finalmente, solo resta considerar las multiplicaciones por factores de rotación ω_{2N}^j. Los casos $j = 0, N/2$ son triviales y los casos $j = N/4$ y $3N/4$ resultan ser multiplicaciones por raíces octavas de la unidad. Antes hemos notado que la multiplicación por una raíz n-ésima de la unidad, cuando $n \in \mathbb{N}\backslash\{1, 2, 4, 8\}$, se puede realizar con 3 sumas reales y 3 multiplicaciones reales. Por tanto, en total, las N multiplicaciones complejas por estos factores de rotación supondrían realizar $3 \cdot 2^t - 8$ sumas y $3 \cdot 2^t - 8$ multiplicaciones reales, considerando que cada multiplicación compleja necesita 3 sumas y multiplicaciones reales, con $N = 2^t$ el número total de multiplicaciones complejas y de sumas complejas. A esto habría que restar 8 operaciones en cada caso: 6 asociadas a los casos triviales $j = 0$ y $j = N/2$, y otras dos correspondientes a multiplicaciones por raíces octavas de la unidad ($j = N/4$ y $j = 3N/4$), donde el número de sumas reales a realizar es 2, igual que el de multiplicaciones complejas. Resumiendo, tenemos que

$$\begin{aligned} \mu(t+1) &= 2 \cdot \mu(t) + 3 \cdot 2^t - 8, \\ \alpha(t+1) &= 2 \cdot \alpha(t) + 2 \cdot 2^{t+1} + 3 \cdot 2^t - 8 \;=\; 2 \cdot \alpha(t) + 7 \cdot 2^t - 8. \end{aligned}$$

Se nos plantean entonces dos ecuaciones en diferencias lineales con coeficientes constantes de primer orden. En general, una ecuación en diferencias lineal de orden n con coeficientes constantes $a_j \in \mathbb{R}$, con $j = 0, 1, \ldots, n-1$ y $a_0 \neq 0$, es una expresión de la forma

$$f(t+n) + a_{n-1} f(t+n-1) + \cdots + a_1 f(t+1) + a_0 f(t) = g(t), \tag{3.9}$$

con $t \in \mathbb{N}$, $g : \mathbb{N} \to \mathbb{R}$ una sucesión dada y $f : \mathbb{N} \to \mathbb{R}$ la sucesión que resuelve la ecuación en diferencias. Introducimos el operador en diferencias

$$Lf(t) = f(t+n) + a_{n-1} f(t+n-1) + \cdots + a_1 f(t+1) + a_0 f(t),$$

y el polinomio característico asociado al operador

$$p(\lambda) = \lambda^n + a_{n-1}\lambda^{n-1} + \cdots + a_1 \lambda + a_0,$$

con $\lambda \in \mathbb{C}$. La solución de (3.9) existe y es única si tenemos como datos iniciales n elementos

consecutivos de la sucesión f. Comenzaremos por buscar la solución del problema homogéneo, es decir,

$$Lf(t) = 0.$$

Tomando $f(t) = \lambda_h^t$ con $\lambda_h \in \mathbb{C}\backslash\{0\}$, entonces

$$L\lambda_h^t \;=\; \lambda_h^{t+n} + a_{n-1}\lambda_h^{t+n-1} + \cdots + a_1\lambda_h^{t+1} + a_0\lambda_h^t \;=\; p(\lambda_h)\lambda_h^t,$$

por lo que $f(t) = \lambda_h^t$ es una solución no trivial del problema homogéneo si y solamente si λ_h es raíz del polinomio característico $p(\lambda)$. Por tanto, la solución general del problema homogéneo es $f(t) = c\,\lambda_h^t$, con $c \in \mathbb{R}$. La solución de la ecuación de (3.9) será una combinación lineal de la solución del problema homogéneo y una solución particular del problema no homogéneo. Calculemos la solución de

$$\mu(t+1) - 2 \cdot \mu(t) = 3 \cdot 2^t - 8. \tag{3.10}$$

Fácilmente, podemos obtener la condición inicial $\mu(2)$. El SFG del algoritmo de Sande-Tukey para una DFT de tamaño 4 se muestra en la Figura 3.6 de abajo.

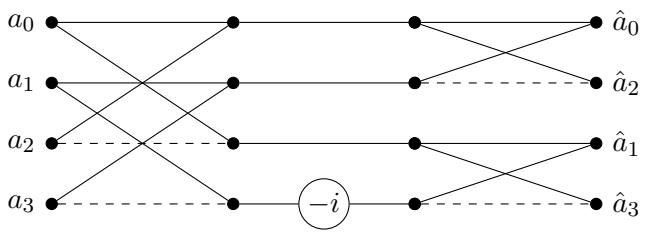


Figura 3.6: Algoritmo de Sande-Tukey para una DFT(4)

Observamos entonces que $\mu(2) = 0$. Con esta condición inicial, la ecuación (3.10) tiene solución única. Atendiendo al método que hemos descrito para el caso general, lo primero que hacemos es calcular la solución de la ecuación homogénea, es decir, la solución de

$$\mu(t+1) - 2 \cdot \mu(t) = 0.$$

En este caso el polinomio característico es

$$p(\lambda) = \lambda - 2,$$

cuya única raíz es $\lambda = 2$. Entonces, para toda constante $c \in \mathbb{R}$, la sucesión $\mu_h(t) = c\,2^t$ es solución del problema homogéneo. Buscaremos una solución particular del problema no homogéneo de la forma

$$\mu_p(t) = c_1\,t\,2^t + c_2. \tag{3.11}$$

Para calcular c_1 y c_2 sustituimos la expresión (3.11) en (3.10) y para que se verifique la igualdad,

debemos tomar $c_1 = \frac{3}{2}$ y $c_2 = 8$. Así, la solución general de (3.10) es

$$\mu(t) = c\,2^t + \frac{3}{2}t2^t + 8,$$

con $c \in \mathbb{R}$. De la condición inicial $\mu(2) = 0$ se sigue que $c = -5$. Entonces, el número de multiplicaciones reales que realiza el algoritmo de Sande-Tukey o el de Cooley-Tukey para calcular una DFT de tamaño N con $N = 2^t$, siendo $t \in \mathbb{N}\backslash\{1\}$, es

$$\mu(t) = -5 \cdot 2^t + \frac{3}{2}t2^t + 8 = \frac{3}{2}N\log_2 N - 5N + 8.$$

Análogamente, resolveríamos la ecuación

$$\alpha(t+1) - 2 \cdot \alpha(t) \;=\; 7 \cdot 2^t - 8. \tag{3.12}$$

El problema homogéneo asociado a esta ecuación diferencial es el mismo que el de la ecuación anterior. Por tanto, su solución general es $\alpha_h(t) = c\,2^t$. Ahora, buscamos una solución del problema no homogéneo de la forma

$$\alpha_p(t) = \tilde{c}_1\, t\, 2^t + \tilde{c}_2,$$

obteniendo $\tilde{c}_1 = \frac{7}{2}$ y $\tilde{c}_2 = 8$. Así, la solución general de (3.12) es

$$\alpha(t) = \tilde{c}\,2^t + \frac{7}{2}\,t\,2^t + 8,$$

con $\tilde{c} \in \mathbb{R}$. Observamos ahora la Figura 3.6 para ver que una DFT(4) requiere realizar 8 sumas complejas, lo que equivale a 16 sumas reales. Es decir, $\alpha(2) = 16$. Teniendo en cuenta este dato inicial, concluimos que el número de sumas reales que hace el algoritmo de Sande-Tukey o el de Cooley-Tukey es

$$\alpha(t) = -5 \cdot 2^t + \frac{7}{2}\,t\,2^t + 8 = \frac{7}{2}N\log_2 N - 5N + 8.$$

Concluimos el apartado con el siguiente teorema, que sintetiza los resultados obtenidos:

Teorema 3.2. *Sea $N = 2^t$ con $t \in \mathbb{N}$ dado. Entonces, el coste computacional de los algoritmos de Sande-Tukey y de Cooley-Tukey para el cálculo de una DFT de tamaño N es igual a*

$$\alpha(t) + \mu(t) = 5N\log_2 N - 10N + 16$$

operaciones aritméticas reales, donde $\alpha(t)$ representa el número de sumas reales y $\mu(t)$ el de multiplicaciones reales.

Capítulo 4

Aplicación: Manipulación de una señal de guitarra

4.1. Introducción

En este capítulo, vamos a usar el material teórico explicado en los capítulos anteriores para desarrollar alguna aplicación de las transformadas de Fourier. El sonido se produce por las vibraciones de los cuerpos. Cuando percibimos un sonido, estamos recibiendo una vibración emitida por algún medio material en forma de movimiento ondulatorio, que penetra por nuestro pabellón auricular y produce una vibración en la membrana del tímpano, transmitiéndose seguidamente por la cadena de huesos del oído medio hacia el interno. Esto llega a nuestro sistema nervioso y nos hace experimentar la sensación sonora. En otras palabras, el sonido es la sensación experimentada cuando ondas producidas por determinados movimientos vibratorios llegan al oído. Cualquier sonido sencillo, como una nota musical, puede describirse en su totalidad especificando tres características de su percepción: el tono, la intensidad y el timbre. Estas características corresponden exactamente a tres características físicas de la onda: su frecuencia, su amplitud y su composición armónica o forma de onda. En esta sección utilizamos como referencia las páginas web [4] y [5].

Los sonidos pueden ser agudos o graves, pero la separación entre unos y otros es muy difusa, ya que siempre tendremos uno más agudo que otro. Un tono equivale a dos semitonos, y ambos sirven para medir la distancia en frecuencia entre dos notas. Así, el nombre de la nota nos define su frecuencia de vibración (medida en ciclos por segundo o hercios) y el intervalo nos especifica la relación que existe entre una nota y otra. Las notas naturales son: Do, Re, Mi, Fa, Sol, La y Si, en orden de más grave a más agudo. Todas se diferencian entre sí un tono, excepto las notas Mi y Fa y las notas Si y Do, cuya distancia es de un semitono. La intensidad es la cantidad de energía acústica que contiene un sonido y viene definida por la potencia, que a su

vez está determinada por la amplitud de la onda sonora, permitiendo distinguir si el sonido es fuerte o débil. Por último, el timbre completa las posibilidades de variedades del arte musical desde el punto de vista acústico. Se trata de una cualidad del sonido que permite distinguir una misma nota musical producida por dos instrumentos musicales diferentes. La explicación de esto se debe a que una misma nota producida por dos instrumentos distintos tiene distinta composición de armónicos. Los armónicos se definen como las componentes de un sonido, que son múltiplos enteros de la frecuencia más baja o fundamental. Esta frecuencia fundamental es lo que habitualmente percibimos como el tono o nota, y por tanto, los armónicos se obtienen al multiplicar la frecuencia de esa nota por un número entero. El Teorema de Fourier plantea que toda señal periódica compleja $f(t)$ puede escribirse como una suma infinita de sinusoides en relación armónica cuya frecuencia fundamental tenga el mismo periodo que la señal compleja que se está analizando, es decir:

$$f(t) = a_0 + \sum_{n=0}^{\infty} [a_n \cos(n\omega_0 t) + b_n \operatorname{sen}(n\omega_0 t)].$$

La expresión anterior no es más que la serie de Fourier en forma real de una función periódica con un cierto periodo T asociado a la frecuencia ω_0. También vimos en el capítulo introductorio que se puede calcular (teóricamente) la transformada de Fourier continua de una función (siempre que la integral tenga sentido), obteniendo así su espectro de frecuencias. Sin embargo, como ya anunciamos, ni la serie de Fourier ni la transformada de Fourier suelen ser fáciles de calcular. Aquí entra en juego la transformada de Fourier discreta, que nos proporciona el contenido frecuencial de una señal solo con que conozcamos el valor de la misma en N puntos. Además, ajustando el número de muestras de la señal para que sea una potencia de 2, podemos utilizar alguno de los algoritmos explicados en la Sección 3.2 para el cálculo óptimo de la DFT.

Una señal de guitarra es una onda continua. Sin embargo, cuando realizamos una grabación con el ordenador, éste no captura la señal de forma continua, sino que realiza un muestreo periódico. Es decir, en vez de registrar los datos de manera continua, se capturan valores en intervalos regulares de tiempo. La frecuencia de muestreo es el número de muestras que se recogen de la señal analógica en un periodo de un segundo. En física, la frecuencia se mide en hercios (Hz). Normalmente, la frecuencia más baja que un oído humano puede captar es 20 Hz y la más alta 20000 Hz. Al realizar un muestreo se desea representar una señal analógica aproximándola mediante una señal digital. Por ende, es importante que el número de muestras por segundo sea suficiente para no perder información. El teorema de muestreo de Nyquist es un resultado fundamental en el campo de la señal: la señal analógica se puede reconstruir a partir de sus muestras siempre que la frecuencia de muestreo sea superior al doble de la frecuencia más alta de la señal. En base a esto, como el ser humano no es capaz de escuchar sonidos con una frecuencia superior a 20000 Hz, la frecuencia de muestreo estándar debería de ser de 40000 Hz. Sin embargo, la mayoría de sistemas de grabación suelen grabar a 44100 Hz. Esto se debe a

un fenómeno conocido como aliasing. El aliasing se produce cuando las frecuencias recogidas más allá del límite establecido no se distinguen o se entienden erróneamente como frecuencias inferiores. Esto puede causar que la señal reconstruida a partir del muestreo en nada se parezca a la señal analógica que se está queriendo recoger en el ordenador. Para evitar la distorsión causada por estas altas frecuencias, las tarjetas de sonido suelen venir con un filtro anti-aliasing en la entrada de la señal, antes de ser convertida a digital. Sin embargo, por razones técnicas, es imposible que los filtros anti-aliasing incorporados a la tarjeta de sonido filtren a partir de la frecuencia inmediatamente superior al rango de audición humano. Por lo tanto, el corte del filtro termina haciendo una curva, disminuyendo gradualmente la entrada de altas frecuencias. Esta curva se llama pendiente. En esta pendiente, el filtro no rechazará ni dejará pasar totalmente todas las frecuencias. Por esta razón, la pendiente del filtro antialiasing debe estar más allá de la frecuencia de 20000 Hz. De lo contrario, generará pérdidas en el sonido que escuchan los humanos. Normalmente, los 44100 Hz son suficientes para proporcionar una zona segura en la que las frecuencias de aliasing y la pendiente del filtro anti-aliasing no afectarán en nada al rango auditivo humano, pero eso depende de la calidad del filtro. El problema es que apenas obtenemos toda la información sobre la calidad de los filtros anti-aliasing de las tarjetas de sonido disponibles en el mercado. Por este motivo, mucha gente prefiere utilizar una alta frecuencia de muestreo, como 88200 Hz, para asegurarse de que el efecto de aliasing, o incluso el filtro anti-aliasing, no interfiera en el contenido de las frecuencias alrededor de 20000 Hz.

Para concluir este trabajo, realizaremos ejemplos prácticos utilizando una señal de guitarra. Para capturar la señal en el ordenador, hemos utilizado una tarjeta de sonido *Focusrite Scarlett Solo* y el software *Ableton* para la grabación. Hemos seleccionado una frecuencia de muestreo estándar de 44100 Hz para asegurar que la señal se captura con la calidad necesaria para el análisis y procesamiento posteriores. El programa *Ableton* se ha utilizado única y exclusivamente para realizar la grabación. La señal que implementamos es el sonido natural de la guitarra, con su correspondiente composición de armónicos y sin realizar ningún tipo de procesamiento digital adicional.

Implementamos la señal en el ordenador mediante la función *audioread*. Esta función tiene un argumento de entrada, que es el archivo que queramos cargar. En nuestro caso, se trata de un archivo *wav* que contiene la grabación de la guitarra, pero serviría cualquier archivo de audio. La función también tiene dos argumentos de salida, que son la señal discretizada, x, y la frecuencia de muestreo, Fs. En el Código 4.1 se muestra el resto de datos que necesitamos.

```
%% Datos
clear all
[x,Fs] = audioread('lates.wav'); % cargamos la senal de audio, Fs es la
  frecuencia de muestreo
x = x(:,1); % convertimos la pista en mono
%%%
% Hacemos que el tamano del vector de entrada sea una potencia de 2
n = 2^(nextpow2(length(x)))/2;
```

```
x = x(1:n);
N = length(x);
%%%
T = N/Fs; % duracion de la pista
dt = 1/Fs; % paso temporal
t = (0:N-1)*dt; % malla temporal
```

Código 4.1: Datos.

Para ver el contenido frecuencial de la señal, podemos utilizar la transformada discreta de Fourier, y más concretamente la FFT. En MATLAB hay un comando, *fft*, que calcula la FFT de cualquier vector dado. También, el comando *ifft* nos proporciona la FFT inversa. Sin embargo, en este trabajo vamos a utilizar el algoritmo de Sande-Tukey (puede verse la implementación en MATLAB en el Apéndice A), explicado en detalle en el capítulo anterior. Para utilizar este algoritmo, es necesario que el vector de entrada tenga una longitud que sea potencia de 2. Si la longitud del vector de entrada no cumple esta condición, recortamos su longitud al valor entero más cercano que sea potencia de 2, truncando el vector original. En el Código 4.1 ya realizamos este ajuste. Lo único que ocurre al proceder de esta manera es que la duración del sonido que estamos procesando en MATLAB será inferior, lo cual carece de importancia para nosotros.

Como la DFT de una señal finita es un vector complejo, para ver gráficamente el contenido frecuencial de la señal, se utiliza el espectro de potencias. Formalmente, el espectro de potencias de una señal $\mathbf{f} \in \mathbb{C}^N$ se define como el vector cuyas componentes son

$$Ps\,\mathbf{f}_k = \hat{\mathbf{f}}_k \overline{\hat{\mathbf{f}}}_k = |\hat{\mathbf{f}}_k|^2,$$

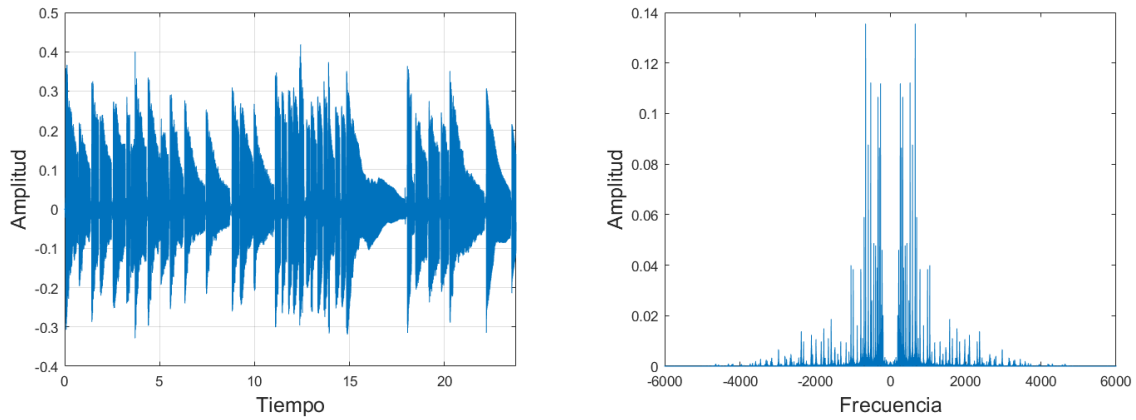
para todo $k = 0, 1, \ldots, N-1$. En 4.2, puede consultarse el código que hemos utilizado en MATLAB para hacer la representación temporal de la señal y la representación frecuencial. Se utiliza una función, *fftfreq*, que fue aportada en la asignatura de *Modelos Matemáticos* (puede consultarse en el Apéndice B).

```
%% Representacion grafica de la senal en el dominio temporal
figure(1);
plot(t,x);
grid on; % Anadir rejilla
xlabel('Tiempo','FontSize',15); % Anadir etiqueta al eje x con tamano de
  fuente 15
xlim([0 T]); % Limitamos el eje x desde 0 hasta la duracion total de la pista
ylabel('Amplitud','FontSize',15);
%%%
%% Representacion de espectro de potencias
X = sande_tukey(x)*dt; % DFT del vector x
freq = fftfreq(N,dt); % malla frecuencial
figure(2);
plot(freq,abs(X)); % representamos
xlim([-6000 6000]) % limitamos el rango de x a 6 kHz, ya que las frecuencias
```

```
% mas alla de este valor no tienen importancia en el sonido de una guitarra
xlabel('Frecuencia','FontSize',15);
ylabel('Amplitud','FontSize',15);
```

Código 4.2: Representación temporal y frecuencial de la señal

En la Figura 4.1a puede verse la representación en el dominio temporal de la señal de guitarra, y en la figura 4.1b la del espectro de potencias. La representación de la señal en el dominio frecuencial nos aporta mucha más información acerca de la misma, ya que podemos observar qué frecuencias son las predominantes y así comprender mucho mejor el sonido que vayamos a escuchar. En el eje de abscisas aparecen las distintas frecuencias, mientras que en el eje de ordenadas se muestran las amplitudes de cada frecuencia. Debido a la periodicidad de la DFT, es indiferente escoger un dominio frecuencial u otro.



(a) **(b)**

Figura 4.1: Representación de la señal en el dominio temporal (izquierda) y en el dominio frecuencial (derecha)

4.2. Un breve paso por la transformada de Fourier ventaneada

Hay ocasiones en las que puede resultar muy interesante representar el tiempo frente a la frecuencia. Esto sucede en particular en el campo de la música, donde las frecuencias de la señal cambian con el tiempo. Sin embargo, debido al principio de incertidumbre de Heisenberg, que es parte de los contenidos de la asignatura *Modelos Matemáticos* (ver también [6]), la transformada de Fourier no es una buena herramienta para este propósito. La herramienta básica para representar información en tiempo y frecuencia simultáneamente es la transformada de Fourier ventaneada. Para entender brevemente en qué consiste, podemos restringirnos al espacio de Hilbert $L^2(\mathbb{R})$. Intuitivamente, lo que se hace con la transformada ventaneada es extraer información frecuencial por intervalos cortos de tiempo. Análogamente, nos interesa la información temporal cerca de

una cierta frecuencia. Esta idea se formaliza mediante el uso de *ventanas*. Una ventana es una función $\omega \in L^2(\mathbb{R})$ par y no negativa. Un ejemplo de ventana muy utilizado en la práctica es la ventana gaussiana de media 0 y varianza σ^2,

$$\omega(t) = \frac{1}{\sigma\sqrt{2\pi}}\, e^{-t^2/2\sigma^2}.$$

Definición 4.1. La transformada ventaneada de Fourier (STFT) de $f \in L^2(\mathbb{R})$ es la función $Sf : \mathbb{R}^2 \to \mathbb{C}$ definida en el plano tiempo-frecuencia por

$$Sf(\tau, \xi) = \int_{\mathbb{R}} f(t)\, \omega(t - \tau)\, e^{-i\xi t}\, dt.$$

Como la transformada de Fourier ventaneada toma valores complejos, para la representación gráfica es necesario recurrir al espectrograma de la señal.

Definición 4.2. Sea $f \in L^2(\mathbb{R})$. Se define el espectrograma de f como la función integrable $f : \mathbb{R}^2 \to \mathbb{R}$ dada por

$$Ps\,(\tau, \xi) = |Sf(\tau, \xi)|^2.$$

También se puede definir una versión discreta de la transformada de Fourier ventaneada (DSTFT), así como del espectrograma. En MATLAB hay una función, *spectrogram*, que calcula el espectrograma de una señal discreta y finita dada. Vamos a ver cómo utilizarla en MATLAB. La función cuenta con numerosos argumentos de entrada, sin embargo, para nuestros propósitos nos interesan, especialmente, seis de ellos. En primer lugar, tenemos que introducir la señal discreta con la que vamos a trabajar. A continuación, introducimos tres parámetros que servirán para ajustar el espectrograma:

- **Tamaño de la ventana (*window*)**: Se refiere a la cantidad de muestras de la señal que se consideran en cada segmento para realizar la Transformada de Fourier. En aplicaciones musicales, se puede tomar un tamaño de 512 ventanas.
- **Número de solapamientos (*noverlap*)**: Es el número de muestras que se comparten entre ventanas consecutivas. Un número de solapamientos mayor hace que veamos más cohesión en el espectrograma. Por otra parte, si el número de solapamientos es pequeño, reducimos el riesgo de perder información entre una ventana y la siguiente, aunque se verán más "saltos" en el gráfico. En nuestras aplicaciones fijaremos este valor en 384.
- **Número de puntos de la FFT (*nfft*)**: Este parámetro determina el número de puntos que se utilizan para calcular la Transformada Rápida de Fourier (FFT) en cada ventana. Al calcular la transformada ventaneada de Fourier discreta, se aplican ventanas de tamaño limitado a la señal original, y en cada ventana se calcula una FFT. Por lo tanto, el número de puntos de la FFT influye en la resolución frecuencial del análisis: cuantos más puntos

seleccionemos, más precisos serán los cálculos pero más costoso será realizarlos. En nuestras aplicaciones, consideraremos un valor de 1024 para este parámetro.

Faltarían dos argumentos de entrada más, que serían la frecuencia de muestreo y otro término, *yaxis*, que lo que hace es indicar que queremos que la frecuencia se represente en el eje de ordenadas.

Por otro lado, la función *spectrogram* cuenta con tres argumentos de salida: el espectrograma de la señal, una malla frecuencial y una malla temporal. En el Código 4.3 se detalla cómo realizar un espectrograma con MATLAB, tras haber declarado los datos del Código 4.1.

```
%% Representamos del espectrograma
window = 512; % Tamano de la ventana de analisis
noverlap = 384; % Numero de muestras de solapamiento
nfft = 1024; % Numero de puntos de la FFT

% Generamos el espectrograma
[S, F, T] = spectrogram(x, window, noverlap, nfft, Fs, 'yaxis');

% Filtramos las frecuencias hasta 6 kHz, que es suficiente para
% analizar el contenido frecuencial de una pista de guitarra
maxFrequency = 6000;
F_idx = F <= maxFrequency;

% Cambiamos la magnitud del espectrograma a decibelios para una mejor
% comprension
S_dB = 10*log10(abs(S(F_idx, :)) + eps);

% Representamos graficamente el espectrograma
figure;
imagesc(T, F(F_idx), S_dB); % Escala logaritmica en dB
axis xy; % Para que el tiempo este en el eje X y la frecuencia en el eje Y
colorbar;

% Ajustamos la escala de color
max_db = max(S_dB(:));
min_db = max_db - 60; % 60 dB por debajo del valor maximo

% Nos aseguramos de que el valor minimo no es menor que el umbral usual de
  ruido
min_db = max(min_db, -100); % No bajar mas de -100 dB, que suele ser el umbral
   de ruido
caxis([min_db, max_db]);

% Anadimos etiquetas
xlabel('Tiempo (s)');
```

```
ylabel('Frecuencia (Hz)');
```

Código 4.3: Espectrograma de la señal de guitarra

Ahora nos centraremos en el análisis gráfico del espectrograma (ver Figura 4.2). Para la representación gráfica del espectrograma estamos usando la escala logarítmica porque consideramos que así la comprensión es más sencilla. La gran ventaja de un espectrograma frente al espectro de potencias es que permite visualizar sobre un mismo gráfico tiempo, frecuencia y amplitud. Esto resulta muy útil, entre otras muchas aplicaciones, para detectar ruidos en la señal, como veremos más adelante.

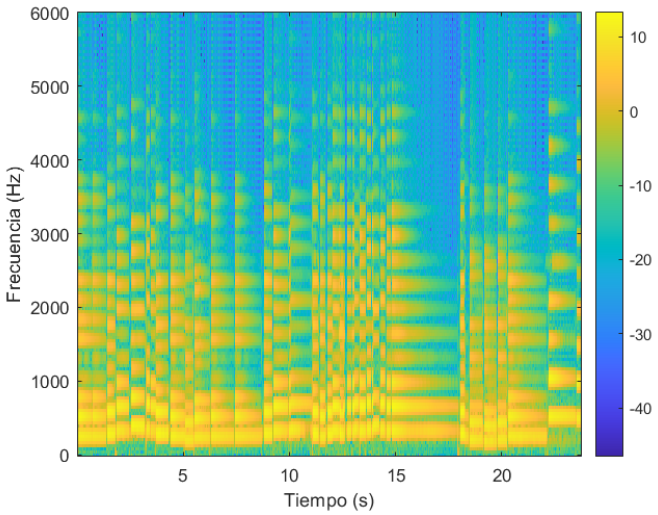


Figura 4.2: Espectrograma de la señal de guitarra. La escala de colores representa la amplitud.

En el eje de abscisas se representa el tiempo, mientras que en el de ordenadas se refleja la frecuencia. La gradación del color en el espectrograma sigue una lógica donde el azul indica una menor energía en las frecuencias dentro del intervalo correspondiente de tiempo, lo que se traduce en una menor intensidad sonora percibida. Por otro lado, tonos más amarillentos indican una mayor amplitud en esas frecuencias. Por tanto, en el gráfico, las áreas con tonos más amarillos representan las frecuencias más destacadas en el sonido de la guitarra, mientras que los tonos más fríos representan armónicos, es decir, múltiplos enteros de la frecuencia fundamental. Estos armónicos son esenciales para definir el timbre de la guitarra, es decir, aquello que caracteriza su sonoridad.

4.3. Creación de un ruido artificial y posterior filtrado

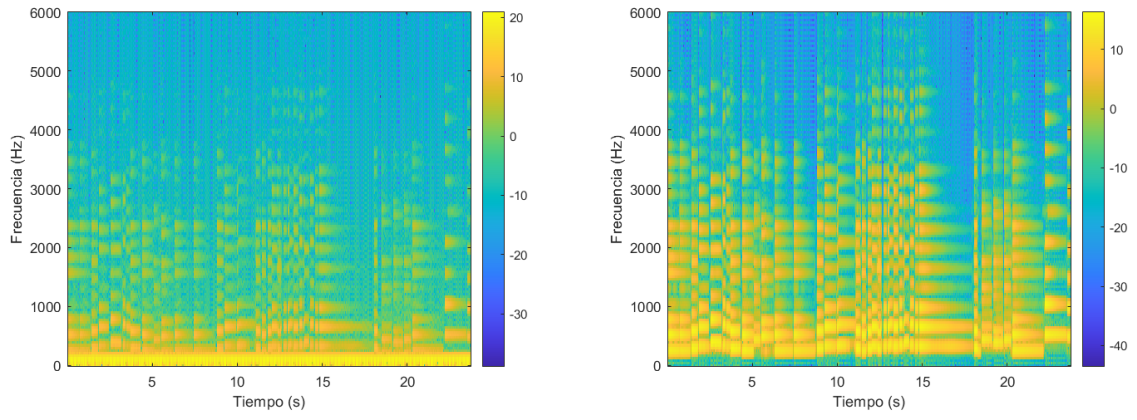
Vamos a comenzar la sección generando un ruido artificial. Hay distintos tipos de ruido: ruido gaussiano, ruido blanco, ruido rosa ... Nosotros vamos a generar un ruido sinosoidal de baja frecuencia. Hay que definir las frecuencias del ruido y sus amplitudes. Si queremos que el ruido sea grave, escogemos frecuencias bajas, por ejemplo: 10, 25, 50 y 100 Hz con amplitudes, respectivamente, de 0.05, 0.1, 0.3, 0.6. En este ejemplo, la amplitud se mide en relación a la amplitud máxima posible en una función sinusoidal, que oscila entre -1 y 1. El Código 4.4 muestra cómo se implementa el ruido en MATLAB.

```
%% Creacion de ruido
% Generamos un ruido artificial
t = (0:length(x)-1) / Fs; % malla temporal
frecuencias = [10, 25, 50, 100]; % Frecuencias bajas para un ruido grave
amplitudes = [0.05, 0.1, 0.3, 0.6]; % Amplitudes para cada frecuencia
ruido = zeros(size(x)); % inicializamos matriz de ceros sobre la que
  generaremos el ruido

%Iteramos ...
for i = 1:length(frecuencias)
ruido = ruido + amplitudes(i) * sin(2 * pi * frecuencias(i) * t)';
end

% Anadimos el ruido generado a la senal de entrada
x_ruidosa = x + ruido;
```

Código 4.4: Generación de ruido artificial y adición a la señal de entrada.



(a) **(b)**

Figura 4.3: Espectrograma de la señal con ruido (izquierda) y tras realizar el filtrado del ruido (derecha).

Podemos escuchar nuestro audio con ruido utilizando el comando de MATLAB *sound*, que tiene dos argumentos de entrada: la señal x y la frecuencia de muestreo Fs. Vamos a analizar el

espectrograma (ver Figura 4.3a) de la nueva señal con ruido para ver el contenido frecuencial a lo largo del tiempo. Lo que más llama la atención en este gráfico es la gruesa franja de color amarillo de la parte inferior. Apenas se distinguen los armónicos de la guitarra. Esto indica que hay frecuencias bajas de alta intensidad sonora, que son precisamente las que generan el ruido en la señal. Tenemos una señal discreta de dimensión finita, *x_ruidosa*, que tiene ruido de frecuencia baja. En el capítulo anterior hemos explicado en detalle la operación de convolución cíclica de vectores. Intuitivamente, al convolucionar dos señales discretas finitas, lo que se hace es mezclarlas. Por tanto, lo que vamos a hacer para eliminar el ruido de baja frecuencia, es convolucionar la señal con ruido con otra señal que atenúe esas bajas frecuencias. Esta señal, que nosotros mismos generamos, es a lo que nos referiremos como filtro frecuencial. Fijaremos una frecuencia de corte de 250 Hz y eliminaremos de la señal todos los sonidos con frecuencias inferiores. Disponemos de una malla temporal, t, de N puntos sobre los que conocemos el valor de *x_ruidosa*. A partir de esta, creamos una malla frecuencial, F, también de N puntos. Elaboraremos un filtro h cuya DFT es el vector $\hat{h}$ con componentes

$$\hat{h}[k] = \begin{cases} 1 & \text{si } F_k > 250 \\ 0 & \text{si } F_k \leq 250, \end{cases}$$

para todo $k = 0, 1, \ldots, N-1$, donde F_k denota el elemento k- ésimo de la malla frecuencial. En virtud del Teorema 2.9, tenemos que la transformada de Fourier discreta de la convolución cíclica de dos señales finitas es igual al producto, punto a punto, de las transformadas de Fourier discretas de cada señal. Por tanto, para aplicar el filtro a la señal con ruido, basta con calcular la DFT de la señal *x_ruido*, multiplicarla punto a punto por el vector $\hat{h}$ y, mediante la DFT inversa, recuperar el resultado en el dominio temporal (ver Código de MATLAB en 4.5)

```
%% Filtrado del ruido
freq_corte = 250; % Hz
dt = 1/Fs; % Paso de tiempo
%%%
F_ruido = sande_tukey(x_ruidosa)*dt; % calculamos el espectro de frecuencias
  de la senal ruidosa y normalizamos el vector
freq_x = fftfreq(length(x_ruidosa), dt)'; % con esta funcion lo que se hace es
   generar una malla frecuencial
h = (abs(freq_x) > freq_corte); % filtro de baja frecuencia
F_ruido_filtrada = 2*F_ruido .* h;  % multiplicamos por 2 para que la amplitud
   sea un poco mas alta, es decir, que se escuche un poco mas fuerte la pista
x_ruido_filtrada = real(sande_tukey_inv(F_ruido_filtrada)/dt); % Conserva solo
   la parte real
```

Código 4.5: Filtrado de ruido

En la Figura 4.3b puede verse el espectrograma de la señal tras aplicar el código anterior. Ahora ya sí que se distinguen las frecuencias fundamentales de cada nota de la guitarra, así como los

distintos armónicos. Si escuchásemos la pista, percibiríamos el sonido limpio de la guitarra.

4.4. Ilustración numérica con matrices de modulación y de desplazamiento positivo

En la Sección 2.4, introdujimos varias matrices y propiedades matriciales. Ahora, nos centraremos en dos matrices: la matriz de modulación y la matriz de desplazamiento positivo. La matriz de modulación $\mathbf{M}_N$, definida por

$$\mathbf{M}_N = \operatorname{diag} \mathbf{e}_1,$$

donde $\mathbf{e}_1$ es el primer vector de la base de vectores introducida en el Teorema 2.1, produce un desplazamiento frecuencial en la señal sobre la que se aplica. Por otro lado, la matriz de desplazamiento positivo,

$$\mathbf{V}_N = \begin{bmatrix} 0 & 0 & \dots & 0 & 1 \\ 1 & 0 & \dots & 0 & 0 \\ 0 & 1 & \dots & 0 & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & \dots & 1 & 0 \end{bmatrix},$$

desplaza hacia la derecha todos los elementos del vector sobre el que actúa. Nuestra intención en esta sección es ilustrar numéricamente la propiedad recogida en el primer apartado de la Proposición 2.8. Puede verse el código de MATLAB en 4.6. Implementamos las matrices en formato *sparse* porque estamos trabajando con un número de puntos muy elevado y además las matrices que implementamos tienen muchos ceros.

```
%% Datos
clear all % limpiamos el espacio de trabajo
%%%
[x,Fs] = audioread('lates.wav'); %leemos la senal de audio
x = x(:,1);
%%
dt = 1/Fs; % calculamos el paso temporal
%%%

% Si asi lo queremos, escuchamos la senal
% sound(x,Fs);

% Potencia de 2 mas cercana por abajo al tamano del vector
N = 2^(nextpow2(length(x)))/2;
x = x(1:N);
X = sande_tukey(x)*dt; % DFT normalizada
%%
```

```
% Primer vector de la base ortogonal introducida en el Teorema 2.1
e1 = exp(-2*pi*1i.*(0:N-1)/N)';
% Matriz de modulacion vista en teoria
MN = spdiags(e1,0,N,N); % la introducimos como matriz sparse debido a que su
  tamano es muy grande

% Matriz de desplazamiento positivo:
VN = spdiags(ones(N-1,1),-1,N,N); % formato sparse
VN(1,N) = 1;
%%%
%% Comprobacion numerica del primer apartado de la Proposicion 2.8
% Multiplicamos matriz de modulacion por DFT de x
X_mod = MN*X;
X_mod = full(X_mod); %convertimos el vector de sparse a completo
%%%
% Multiplicamos la matriz de desplazamiento positivo por el vector x
x_mod = VN*x;
x_mod = full(x_mod); %convertimos el vector de sparse a completo

%%%
% Multiplicar la matriz de desplazamiento positivo por el vector x, y
% calcular la DFT del resultado, deberia de ser equivalente a
% multiplicar la matriz de modulacion por la DFT del vector x
%%%
% Comprobaremos que esto es asi representando los espectros de potencias en
% ambos casos
%%%
dftx_mod = sande_tukey(x_mod)*dt;
freq = fftfreq(N,dt); % malla frecuencial

% Representamos a la vez los espectros de frecuencias de las dos senales y
% vemos que coinciden
figure;
plot(freq,abs(X_mod));
xlim([-6000 6000]);
xlabel('Frecuencia','FontSize',15);
ylabel('Amplitud','FontSize',15);
figure;
plot(freq,abs(dftx_mod));
xlim([-6000 6000]);
xlabel('Frecuencia','FontSize',15);
ylabel('Amplitud','FontSize',15);
```

Código 4.6: Matriz de modulación y matriz de desplazamiento positivo

En la Figura 4.4 se muestra la gráfica correspondiente al espectro de potencias obtenido en ambos casos. Así, comprobamos con un ejemplo básico que la identidad matricial del primer apartado de la Proposición 2.8 es cierta. Recordamos que, lo que en el fondo nos dice la proposición en

cuestión, es que resulta equivalente desplazar la señal en el dominio temporal o en el frecuencial, en el sentido de es indiferente mover los elementos del vector hacia la derecha en el dominio temporal, o hacer un desplazamiento en términos de frecuencia mediante un cambio de fase.

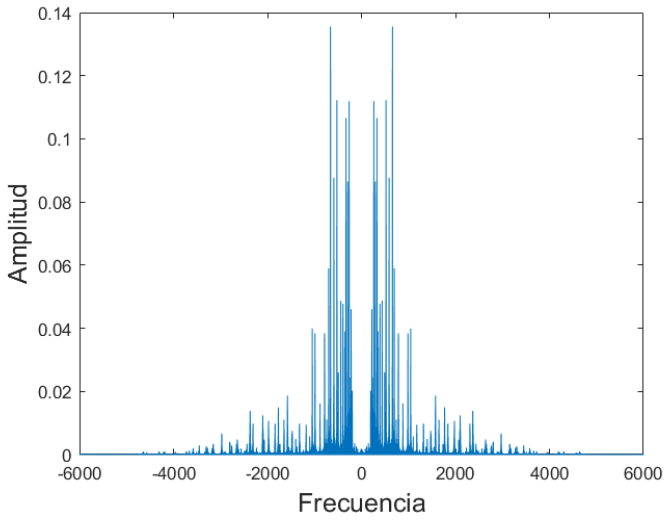


Figura 4.4: Espectro de potencias idéntico en ambos casos

4.5. Creación de un efecto de *delay*

Seguramente más de uno ha disfrutado de un vibrante concierto de rock, blues, pop o cualquier otro género, donde la guitarra eléctrica toma el escenario. ¿Y quién no se ha preguntado alguna vez por qué el guitarrista parece estar en una guerra de baile con el suelo, dando patadas aquí y allá? Como guitarrista, he sido testigo de estas curiosas preguntas del público al finalizar un concierto, e incluso mis compañeros de banda han bromeado sobre mis "dotes de bailarín" al darle golpecitos a mi querida pedalera de efectos.

Pero, ¿cuál es la verdad detrás de estos movimientos aparentemente extravagantes? La realidad es que cada "patada" es un toque de magia, un gesto que transforma un simple acorde en una explosión de sonido que reverbera en el alma de cada espectador. ¿Recuerdas la inconfundible guitarra de Brian May en el himno universal *We Are the Champions*?

Para aquellos que no están familiarizados con el mundo del rock, la pedalera de efectos es como una caja de sorpresas llena de posibilidades sonoras infinitas. Un pisotón acá, y de repente estás sumergido en un mundo de 'wah-wah' funky al más puro estilo *Red Hot Chili Peppers*. Otro pisotón allá, y te encuentras flotando en un océano de eco que te lleva a otra dimensión. Cuando ves a un guitarrista saltar y golpear con pasión el suelo, es como si estuviera invocando a los espíritus del rock para que le ayuden a crear una experiencia única, una canción que al escucharla te recuerde a él.

Pues bien, en esta sección utilizaremos las transformadas de Fourier para aplicar un efecto de *delay* a la guitarra. Este efecto, que crea una sensación de eco, consiste en superponer varias veces la pista de guitarra con un pequeño retardo en el tiempo. Para lograr esto, vamos a trabajar en el espacio de frecuencias, modificando la fase angular de las distintas muestras de la señal.

Primero, calculamos la Transformada de Fourier Discreta (DFT) de la señal de guitarra. Luego, iniciamos un bucle con un número de iteraciones igual al número de pistas superpuestas que deseamos crear. En cada iteración, multiplicamos punto a punto la DFT de la señal de guitarra por un factor de fase correspondiente. Esta operación, similar al filtrado de ruido, equivale a convolucionar la señal de guitarra con cada factor de fase en el dominio del tiempo.

A continuación, aplicamos la Transformada de Fourier Inversa (IDFT) a cada una de estas nuevas señales en el espacio de frecuencias para devolverlas al dominio del tiempo. Finalmente, sumamos todas estas señales para obtener el efecto de *delay* deseado (ver Código 4.7).

```
function  x_mezcla = delay(x,Fs,T)
% Funcion que aplica un efecto de delay a la senal de audio de entrada
% Parametros de entrada:
%    - x: senal de audio; tiene que ser leida previamente mediante
%       audioread u otra funcion similar
%    - Fs: frecuencia de muestreo
%    - T: parametro de delay. Puede ser un numero o un vector, en funcion
%         del numero de muestras de delay que se quieran. Cada elemento del
%         vector, indica el retardo de cada muestra
% Parametros de salida:
%    - Senal con el efecto de delay

dt = 1/Fs;

% Numero de senales con retardo de fase que se crean a partir de la senal
% original
muestras_delay = length(T);

% Aplicar el efecto de delay en el dominio de la frecuencia
N = length(x); % Longitud de la senal
f = fftfreq(N, dt); % Malla frecuencial
X = sande_tukey(x); % fft del vector x

X_delayed = zeros(N, muestras_delay);
x_delayed = X_delayed;
for i = 1:muestras_delay
% Calcular el factor de fase para el retardo deseado
factor_fase = 1/(4^i)*exp(-1i*2*pi*f*T(i));
% Aplicamos el efecto de delay en el dominio de la frecuencia,
% mediante una convolucion de la senal X y un retardo de fase
X_delayed(:, i) = X .* factor_fase.';
x_delayed(:, i) = sande_tukey_inv(X_delayed(:,i));
```

```
end

% Sumamos las senales con delay a la senal original, ya en el dominio
% temporal
x_mezcla = x + sum(x_delayed, 2);
end
```

Código 4.7: Creación de un efecto de delay

```
x_mezcla = delay(x,Fs,[0.6 1.2 1.8 4]);
sound(x_mezcla,Fs)
```

Código 4.8: Ejemplo de uso de la función *delay*

El abanico de posibilidades para procesar señales de audio utilizando transformadas de Fourier es inmenso. Hemos contemplado también la implementación de otros efectos, como por ejemplo la distorsión, ese efecto de sonido que hace que la guitarra suene tan agresiva, característico de bandas de hard rock o heavy, como ACDC. También el chorus, que crea una sensación de que hay varias guitarras sonando a la vez; o el wah-wah, muy utilizado en géneros como el funk, ejemplificado en la canción *Voodoo Child*, de Jimmy Hendrix.

Sin embargo, por razones de tiempo y espacio es imposible abarcar todo. Por ello hemos querido dar un marco general de qué tipo de cosas se pueden hacer usando estas herramientas. Este trabajo busca aportar una base sólida del análisis de señales discretas, sobre la cual se puedan construir aplicaciones más complejas, permitiendo la exploración y desarrollo de nuevas técnicas en el procesamiento de señales de audio.

Apéndice A

Implementación del algoritmo de Sande-Tukey en MATLAB

```
function X = sande_tukey(x)
  % Funcion que calcula la DFT de un vector de tamano una potencia de 2
  % mediante el algoritmo de Sande-Tukey
  % Argumentos de entrada: vector x de tamano una potencia de 2
  % Argumentos de salida: DFT del vector x

  N = length(x); % tamano del vector de entrada

  % Verificamos si N es potencia de 2, en caso contrario error
  if log2(N) ~= floor(log2(N))
    error('El tamano del vector de entrada debe ser una potencia de 2.');
  end

  t = log2(N); % N = 2^t

  % Reordenamos los indices segun el orden de bits inverso
  x = bitrevorder(x);

  % Comenzamos a iterar
  X = x;
  for etapa = 1:t
    M = 2^etapa; % Tamano de las sub-DFT
    W_M = exp(-1i * 2 * pi / M); % Factor de rotacion

      for k = 0:N/M-1
        for j = 0:M/2-1
          u = X(k*M + j + 1);
          v = X(k*M + j + M/2 + 1) * W_M^j;
          X(k*M + j + 1) = u + v;
          X(k*M + j + M/2 + 1) = u - v;
```

```
        end
      end
  end
end
```

Código A.1: Algoritmo de Sande-Tukey para calcular la DFT de un vector de tamaño una potencia de 2.

```
function x = sande_tukey_inv(X)
  % Funcion que calcula la DFT inversa de un vector mediante el algoritmo de
  % Sande-Tukey
  % Argumentos de entrada: el vector X de tamano una potencia de 2
  % Argumentos de salida: la DFT inversa del vector X

  N = length(X); % tamano del vector de entrada

  % Verificamos si N es potencia de 2, en caso contrario error
  if log2(N) ~= floor(log2(N))
    error('El tamano del vector de entrada debe ser una potencia de 2.');
  end

  t = log2(N); % N = 2^t

  % Reordenamos los indices segun el orden de bits inverso
  X = bitrevorder(X);

  % Comenzamos a iterar
  for etapa = 1:t
    M = 2^etapa; % Tamano de las sub-DFT
    W_M = exp(1i * 2 * pi / M); % Factor de rotacion (conjugado)

      for k = 0:N/M-1
        for j = 0:M/2-1
          u = X(k*M + j + 1);
          v = X(k*M + j + M/2 + 1) * W_M^j;
          X(k*M + j + 1) = u + v;
          X(k*M + j + M/2 + 1) = u - v;
        end
      end
  end

  % Hay que dividir el resultado por N
  x = X / N;
end
```

Código A.2: Algoritmo de Sande-Tukey para calcular la DFT inversa de un vector de tamaño una potencia de 2.

Apéndice B

Implementación de la función que crea la malla frecuencial

```
function freq = fftfreq(N, dt)
  % fftfreq calcula las frecuencias asociadas a las muestras de una transformada
     de Fourier.
  % Argumentos de entrada:
  %   - N: numero de puntos de datos de la senal
  %   - dt: intervalo de tiempo entre cada punto de datos
  % Argumento de salida:
  %   - freq: vector de frecuencias

  % Convertimos dt y N a tipo de datos double para evitar problemas de division
  dt = double(dt);
  N = double(N);

  % Verificamos si N es impar o par para manejar la simetria de la FFT
  if mod(N,2) == 1
    % Si N es impar
    % Frecuencias positivas desde 0 hasta (N-1)/2
    freq1 = (0:(N-1)/2)/(dt*N);
    % Frecuencias negativas desde -(N-1)/2 hasta -1
    freq2 = (-(N-1)/2:1:-1)/(dt*N);
  else
    % Si N es par
    % Frecuencias positivas desde 0 hasta (N/2)-1
    freq1 = double((0:(N/2)-1)/(dt*N));
    % Frecuencias negativas desde -N/2 hasta -1
    freq2 = double((-N/2:1:-1)/(dt*N));
  end

  % Resultado
  freq = [freq1, freq2];
```

```
end
```

Código B.1: Malla frecuencial

Bibliografía

[1] Apostol, T.M. *Análisis matemático*. Reverté, 2020.

[2] Ash, R.B. *Basic probability theory*. Courier Corporation, 2008.

[3] Folland, G.B. *Fourier analysis and its applications*, tomo 4. American Mathematical Soc., 2009.

[4] luz-wiki contributors. La música a través del análisis de fourier. https://luz.izt.uam.mx/wikis/mediawiki/index.php?title=La_m%C3%BAsica_a_trav%C3%A9s_del_an%C3%A1lisis_de_Fourier&oldid=33138, 2023. Fecha de consulta: 2024-05-24.

[5] Magroove. Frecuencia de muestreo : ¿cuál usar? ¿cuál es la mejor? https://magroove.com/blog/es-mx/frecuencia-de-muestreo/, 2020. Fecha de consulta: 2024-05-24.

[6] Plonka, G., Potts, D., Steidl, G. y Tasche, M. *Numerical fourier analysis*. Springer, 2018.

[7] Proakis, J.G. y Manolakis, D.G. *Digital signal processing (3rd ed.): principles, algorithms, and applications*. Prentice-Hall, Inc., USA, 1996. ISBN 0133737624.